UNIVERSIDAD NACIONAL
AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

Una demostración del segundo teorema de Gödel a través de complejidad Kolmogórov

T E S I S

QUE PARA OBTENER EL TÍTULO DE

MATEMÁTICO

P R E S E N T A

RICARDO JAIMES URBÁN

ASESOR DE TESIS
DR. CARLOS TORRES ALCARAZ

CD. DE MÉXICO 2021

Introducción

En 1931 Gödel publicó un artículo titulado *Sobre proposiciones formalmente indecidibles en Principia Mathematica y sistemas afines I* [4], en el cual enuncia sus teoremas de incompletud y demuestra el primero de ellos. Dichos teoremas imponen fuertes limitaciones para cualquier teoría axiomática de primer orden que contenga a la aritmética. El primero de ellos establece que si la teoría es consistente entonces es incompleta, es decir, existen enunciados de su lenguaje tales que ni ellos ni sus negaciones pueden deducirse a partir de los axiomas. Por otro lado, el segundo de ellos afirma que, en particular, el enunciado que expresa la consistencia de la teoría no es demostrable dentro de ella.

La demostración del primer teorema de Gödel puede resultar muy compleja, debido a que recurre a una autorreferencia inspirada en las siguientes paradojas:

- Paradoja de Richard: Algunas frases del idioma español definen propiedades de los números naturales. Si bien la lista de estas expresiones es infinita, cada frase definitoria está compuesta por un número finito de caracteres. Debido a ello, tales definiciones pueden ordenarse conforme a su longitud y las de la misma longitud, alfabéticamente. Con base en esta numeración, definimos que un número es *richardiano* si y sólo si no tiene la propiedad que numera. Puesto que esta definición está formulada en español, le corresponde algún número r. Notemos que entonces surge la paradoja "r es richardiano si y sólo si no es richardiano". Si r fuera richardiano, la propiedad que numera no le correspondería, por lo que no sería richardiano. Análogamente, si r no fuera richardiano, tendría la propiedad que numera, por lo que sería richardiano.

- Paradoja del mentiroso: La versión moderna de esta paradoja se enuncia comúnmente como "Este enunciado es falso". Notemos que si dicho

> enunciado fuera verdadero, entonces sería falso. Análogamente, si fuera falso, entonces sería verdadero, pues sería falso que es falso. Por lo tanto, el enunciado es verdadero si y sólo si es falso.

Esto motivó a matemáticos como Boolos y Chaitin [2] a dar pruebas más simples inspiradas en la paradoja de Berry, que se enuncia como sigue:

Consideremos el enunciado "El menor entero positivo que no se puede definir con menos de quince palabras" y N el menor entero positivo que no se puede definir con menos de quince palabras. Notemos entonces, que el enunciado anterior cuenta con catorce palabras y define al número N, por lo que tenemos una contradicción, pues hemos definido con catorce palabras al menor entero positivo que no se puede definir con menos de quince palabras.

Pese a que el segundo teorema nunca fue demostrado por Gödel, sí fue probado por Hilbert y Bernays en *Grundlagen der Mathematik*, Torres en *Limitaciones internas de la aritmética formalizada* [18], Kikuchi en *A note on Boloos' Proof of the Incompleteness Theorem* [7] y en *Kolmogorov complexity and the second incompleteness theorem* [8], etc. Sin embargo, la mayoría de estas pruebas son muy complejas, pues algunas de ellas recurren a la teoría de modelos.

En este trabajo se presenta una prueba relativamente simple del segundo teorema de Gödel, inspirada en la prueba dada por Kritchman y Raz en 2010, a través del teorema de Chaitin. Este teorema básicamente nos dice que si la aritmética es consistente, entonces es posible encontrar una constante c tal que para todo número natural x no se puede demostrar (dentro de AR) que tenga complejidad Kolmogórov mayor que c.

El trabajo se estructura de la siguiente manera:

En el capítulo 1 explicamos brevemente el concepto de función recursiva. Posteriormente, definimos y explicamos detalladamente los conceptos de máquina de Turing y máquina de Turing universal.

En el capítulo 2 definimos los conceptos de consistencia, ω-consistencia y correctud de una teoría matemática. Asimismo, hacemos notar sus diferencias.

En el capítulo 3 definimos el concepto de complejidad Kolmogórov, la cual forma parte esencial del teorema de Chaitin.

En el capítulo 4 definimos el concepto de completud, discutimos brevemente el primer teorema de Gödel y demostramos el teorema de Chaitin.

En el capítulo 5 analizamos brevemente una de las pruebas del segundo teorema de Gödel dadas por Kikuchi y, por último, demostramos el segundo teorema de Gödel mediante un argumento que involucra al teorema de Chaitin.

Convenciones y notaciones

En esta tesis asumimos que el lector tiene conocimientos básicos sobre teoría de las funciones recursivas, teoría matemática de la computación, teoría de modelos y teoría de conjuntos. Asimismo, usamos las siguientes convenciones y notaciones, con base en el libro *Limitaciones internas de la aritmética formalizada. Un estudio con especial énfasis en el segundo teorema de Gödel* [18].

- Dada una teoría T, L_T representa su lenguaje.
- Dado un número natural k, sk representa a su sucesor.
- Dado un número natural k, $\bar{k}$ representa su numeral ($\bar{k} = \underbrace{s \ldots s}_{k \text{ veces}} 0$).
- Dada una teoría T y una fórmula $A \in L_T$, $\vdash_T A$ significa que A es un teorema de T.
- Dados una teoría T, $x \in L_T$ y $\mathfrak{U}$ una estructura de T, $(x)^{\mathfrak{U}}$ denota la interpretación de x en $\mathfrak{U}$.
- AR representa a la aritmética recursiva de primer orden.
- Dada una propiedad recursiva $P(x)$, $\mu x P(x)$ representa el menor x que satisface $P(x)$. Se trata de una función parcial, ya que sólo está definida cuando dicho número existe.
- Dada una fórmula A, γA representa su número de Gödel.
- Dada una fórmula A, $\ulcorner A \urcorner$ representa $\overline{\gamma A}$.
- Dados los números de Gödel de dos expresiones x e y, $x * y$ representa el número de Gödel de su concatenación.

- Dada S una sucesión de fórmulas, $\gamma^2 S$ representa su número de secuencia.
- Dado el número de secuencia de una sucesión de fórmulas x, $PRUEBA(x)$ significa "x es una prueba".
- Dados el número de Gödel de una fórmula x y el número de secuencia de una sucesión de fórmulas y, $PR(y, x)$ significa "y es una prueba de x".
- Dadas una relación recursiva R y una función recursiva f, $\bar{R}$ y $\bar{f}$ denotan sus *denominaciones* respectivamente. Es decir, las expresiones formales correspondientes a ellas en L_{AR}.
- Dado el número de Gödel de una fórmula, x_0, definimos $TEO(x_0)$ como $TEO(x_0) \equiv_{def} \exists x_1 \overline{PR}(x_1, x_0)$.
- Dada una máquina de Turing M, ε representa la palabra "vacía" de su alfabeto.
- Dada una máquina de Turing M, $M(x)\uparrow$ significa que la máquina M aplicada a x diverge (no se detiene o se detiene sin computar nada). Cuando $x = \varepsilon$, simplemente escribimos $M\uparrow$.
- Dada una máquina de Turing M, $M(x)\downarrow$ significa que la máquina M aplicada a x converge (se detiene habiendo computado algo). Si $M(x)$ converge a y escribimos $M(x)\downarrow y$. Cuando $x = \varepsilon$, simplemente escribimos $M\downarrow y$.
- Dadas dos máquinas de Turing M y M', denotamos que M es equivalente a M' con $M \simeq M'$.
- Dado un conjunto X, $Card(X)$ representa su cardinalidad.

Índice general

Introducción VII

Convenciones y notaciones IX

1. Funciones recursivas y máquinas de Turing 1
- 1.1. Funciones recursivas . . . 1
  - 1.1.1. Definición . . . 1
  - 1.1.2. Clasificación y ejemplos . . . 3
- 1.2. Máquinas de Turing . . . 4
  - 1.2.1. Idea Intuitiva . . . 4
  - 1.2.2. Definición formal . . . 5
  - 1.2.3. La máquina universal $\mathcal{U}$ . . . 8
    - 1.2.3.1. Subrutinas . . . 8
    - 1.2.3.2. Descripciones estándar y números de descripción . . . 17
    - 1.2.3.3. Descripción de $\mathcal{U}$ . . . 18

2. Consistencia, ω-consistencia y correctud 31
- 2.1. Modelos de la aritmética . . . 31
  - 2.1.1. Algunos conceptos, nociones y resultados sobre modelos y números ordinales . . . 32
    - 2.1.1.1. Algunos conceptos de teoría de modelos . . . 32
    - 2.1.1.2. Algunas nociones relativas a los números ordinales . . . 34
  - 2.1.2. Modelo estándar de la aritmética . . . 36

2.1.3. Modelos no estándar de la aritmética 37
2.2. Definiciones, diferencias y similitudes 46

3. Complejidad Kolmogórov **51**
3.1. Idea intuitiva 51
3.2. Definición formal 52

4. Incompletud y consistencia **57**
4.1. Completud 57
4.2. Primer teorema de incompletud de Gödel 57
4.3. Complejidad e Incompletud: El teorema de Chaitin 59

5. Segundo teorema de Gödel **63**
5.1. Idea intuitiva 63
5.2. Demostración dada por Kikuchi 63
5.3. Demostración del segundo teorema de Gödel 66

6. Conclusiones **71**

Bibliografía **73**

Capítulo 1

Funciones recursivas y máquinas de Turing

Como veremos en el capítulo 3, es posible definir el concepto de complejidad Kolmogórov utilizando funciones recursivas, máquinas de Turing o lenguajes de programación. Puesto que optaremos por usar la definición mediante máquinas de Turing, dedicaremos la mayor parte de este capítulo a definir y explicar el funcionamiento de dichas máquinas. Haremos especial énfasis en la definición y el funcionamiento de la máquina universal; pues, como veremos también en el capítulo 3, la definición de complejidad Kolmogórov se basa esencialmente en el funcionamiento de la máquina universal. Previamente, explicaremos de manera breve algunos conceptos de la teoría de funciones recursivas que nos serán de utilidad en las demostraciones de diversos teoremas de los capítulos 4 y 5.

1.1. Funciones recursivas

El concepto de función recursiva nace de la necesidad de hacer más preciso el concepto de *función calculable*. En esta sección presentamos una lista de funciones recursivas iniciales y reglas básicas para generar nuevas funciones recursivas. Asimismo, describimos brevemente cómo se clasifican de acuerdo con su construcción y damos algunos ejemplos.

1.1.1. Definición

Funciones iniciales

Las funciones iniciales se aceptan como inmediatamente calculables. Son las siguientes:

1. Función sucesor $s : \mathbb{N} \to \mathbb{N}$ de grado[1] 1, que a cada número natural x le asigna el que le sigue en la sucesión numérica, denotado con sx.

2. Función constante cero $K_{0,0}$ de grado 0.

3. Función proyección k, $P_{n,k} : \mathbb{N}^n \to \mathbb{N}$ de grado n, que a cada n-ada de números naturales les asigna su k-ésima coordenada. Es decir, $P_{n,k}(x_1, \ldots, x_n) = x_k$.

Reglas para generar nuevas funciones

A partir de las funciones iniciales, es posible generar nuevas funciones recursivas mediante las siguientes reglas:

1. Composición: Si g es una función de grado $m > 0$ y $h_1, \ldots, h_m$ son m funciones de grado n, entonces la ecuación

$$f(x_1, \ldots, x_n) = g(h_1(x_1, \ldots, x_n), \ldots, h_m(x_1, \ldots, x_n))$$

   define una función de grado n.

2. Recursión: Si g es una función de grado n y h es una función de grado $n + 2$, entonces las ecuaciones:

$$\begin{aligned} f(x_1, \ldots, x_n, 0) &= g(x_1, \ldots, x_n) \\ f(x_1, \ldots, x_n, sy) &= h(x_1, \ldots, x_n, y, f(x_1, \ldots, x_n, y)) \end{aligned}$$

   definen una función de grado $n + 1$.

3. Minimalización: Si g es una función de grado $n+1$ con la propiedad de que para cada n números $k_1, \ldots, k_n$ hay un k tal que $g(k_1, \ldots, k_n, k) = 0$, entonces la ecuación

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

[1] Con *grado* nos referimos al número de argumentos de la función.

define una función de grado n. A μ se le llama operador minimal.

Nota: Esta definición suele utilizarse en un sentido más general, eliminando la exigencia de que exista un cero para cada colección $k_1, \ldots, k_n$, obteniendo así una función recursiva parcial. Eliminar dicha exigencia resulta conveniente, pues las máquinas de Turing no necesariamente son totales, por lo que si usáramos la definición anterior, varias máquinas de Turing no tendrían una función asociada[2].

1.1.2. Clasificación y ejemplos

Clasificación

Decimos que una función es recursiva si es inicial o se genera a partir de las funciones iniciales aplicando las reglas anteriores un número finito de veces. Si en el proceso no se aplica la regla de minimalización, entonces decimos que la función es *recursiva primitiva.*

Ejemplos

A continuación mostramos algunos ejemplos de funciones recursivas con el propósito de hacer más claras las definiciones anteriores.

Ejemplo 1.1. Función suma. Consideremos las funciones $P_{1,1}, P_{3,3}$ y s. La función $+(x, y)$ se define como:

$$\begin{aligned} +(x, 0) &= P_{1,1}(x) \\ +(x, sy) &= s(P_{3,3}(x, y, +(x, y))) \end{aligned}$$

y se construye como sigue:

1. $s(x)$ inicial
2. $P_{1,1}(x)$ inicial
3. $P_{3,3}(x, y, z)$ inicial
4. $s(P_{3,3}(x, y, z))$ composición 1 y 3
5. $\left.\begin{aligned} +(x, 0) &= P_{1,1}(x) \\ +(x, sy) &= s(P_{3,3}(x, y, +(x, y))) \end{aligned}\right\}$ recursión 2 y 4

[2]Decimos que una función φ es la función asociada a una máquina de Turing $\mathcal{T}$ si sus dominios coinciden y $\mathcal{T}(x)\!\downarrow\varphi(x)$ para toda x en su dominio.

Ejemplo 1.2. Función producto. Consideremos las funciones $K_{0,0}, P_{3,3}, P_{3,1}$ y $+$. La función $\cdot(x,y)$ se define como:

$$\begin{aligned}\cdot(x,0) &= K_{1,0}(x)\\ \cdot(x,sy) &= +(P_{3,1}(x,y,\cdot(x,y)), P_{3,3}(x,y,\cdot(x,y)))\end{aligned}$$

y se construye como sigue:

1. $s(x)$ inicial
2. $P_{1,1}(x)$ inicial
3. $P_{3,3}(x,y,z)$ inicial
4. $s(P_{3,3}(x,y,z))$ composición 1 y 3
5. $\left.\begin{aligned}+(x,0) &= P_{1,1}(x)\\ +(x,sy) &= s(P_{3,3}(x,y,+(x,y)))\end{aligned}\right\}$ recursión 2 y 4
6. $K_{0,0}$ inicial
7. $P_{2,2}(x,y)$ inicial
8. $\left.\begin{aligned}K_{1,0}(0) &= K_{0,0}\\ K_{1,0}(sx) &= P_{2,2}(x, K_{1,0}(x))\end{aligned}\right\}$ recursión 6 y 7
9. $P_{3,1}(x,y,z)$ inicial
10. $+(P_{3,1}(x,y,z), P_{3,3}(x,y,z))$ composición 3, 5 y 9
11. $\left.\begin{aligned}\cdot(x,0) &= K_{1,0}(x)\\ \cdot(x,sy) &= +(P_{3,3}(x,y,\cdot(x,y)), P_{3,1}(x,y,\cdot(x,y)))\end{aligned}\right\}$ recursión 8 y 10

1.2. Máquinas de Turing

Los conceptos de máquina de Turing y máquina de Turing universal, fueron introducidos por Alan M. Turing en 1936 en el artículo titulado *On computable numbers, with an application to the Entscheidungsproblem* [19]. En esta sección explicaremos y ejemplificaremos tales conceptos.

1.2.1. Idea Intuitiva

Intuitivamente, una máquina de Turing es una máquina imaginaria que cuenta con:

- una cinta infinita dividida en celdas, dentro de las que se escriben y manipulan distintos símbolos pertenecientes a un alfabeto;
- un cabezal de lectura/escritura capaz de inspeccionar una celda a la vez y de desplazarse hacia la izquierda o hacia la derecha una celda a la vez, y
- una unidad de control que se encuentra en un cierto estado q perteneciente a un conjunto finito de estados $\{q_1, \ldots, q_n\}$.

El comportamiento de esta máquina, representada en la figura 1.1, será determinado a partir del estado en el que se encuentre la unidad de control y del símbolo sobre la celda escaneada, mismo que podrá consistir en modificar el símbolo escrito sobre la cinta, mover el cabezal hacia la izquierda o hacia la derecha y cambiar el estado de la unidad de control.

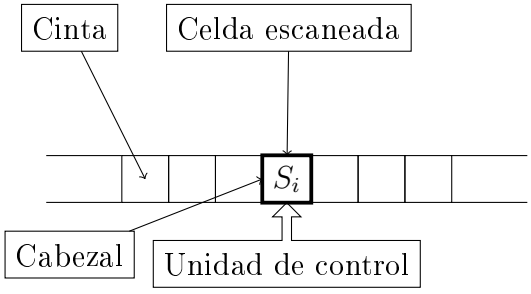


Figura 1.1: Una máquina de Turing

1.2.2. Definición formal

Formalmente una máquina de Turing se define como sigue:

Sean X, Y alfabetos tales que $X \subset Y$, y $\sqcup$ el símbolo "blanco" de $Y \backslash X$. Una máquina de Turing con alfabeto de cinta Y y alfabeto de entrada X es una tétrada $\mathcal{M} = (\mathcal{Q}, \delta, q_0, q_f)$ formada por:

- un conjunto finito de estados $\mathcal{Q}$,

- una función parcial $\delta : \mathcal{Q} \times Y \to Y \times \{R, L, N\} \times \mathcal{Q}$,
- un estado inicial $q_0 \in \mathcal{Q}$, y posiblemente
- un estado final $q_f \in \mathcal{Q}$[†],

donde $\delta(q_f, y)$ no está definida para toda $y \in Y$, y los símbolos R, L, N se interpretan como *moverse a la derecha, moverse a la izquierda* y *no moverse* respectivamente.

Usualmente las máquinas de Turing se describen mediante la lista de asignaciones de su función de transición como sigue:

$$\begin{array}{ccc} \delta(q,s) = (s', m, q') & \text{ó} & (q,s) \mapsto (s', m, q') \\ \vdots & & \vdots \end{array}$$

Sin embargo, nosotros las describiremos mediante tablas como la que se muestra a continuación, pues esta escritura nos resultará útil más adelante.

Estado	*Símbolo*	*Operaciones*	*Estado final*
q	s	$Ps'/E, m$	q'
$\vdots$	$\vdots$	$\vdots$	$\vdots$

En esta tabla Ps' indica "imprimir" el símbolo s' sobre la celda escaneada, E indica borrar la celda escaneada y m indica el movimiento que debe realizar el cabezal. A continuación mostramos como ejemplo el funcionamiento de una máquina en particular.

Ejemplo 1.3. Consideremos la máquina $\mathcal{M}$ descrita por la tabla

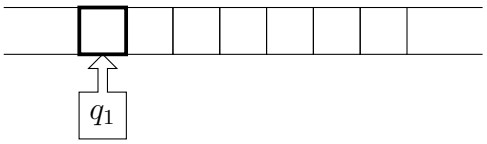
Estado	*Símbolo*	*Operaciones*	*Estado final*
q_1	$\llcorner\!\lrcorner$	$P1, R$	q_2
q_2	$\llcorner\!\lrcorner$	$P0, N$	q_3

cuya función de transición escrita formalmente es:

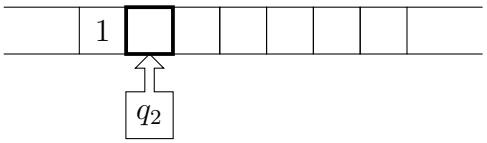
$$\begin{array}{rcl} (q_1, \llcorner\!\lrcorner) & \mapsto & (1, R, q_2) \\ (q_2, \llcorner\!\lrcorner) & \mapsto & (0, N, q_3) \end{array}$$

Su funcionamiento es el siguiente:

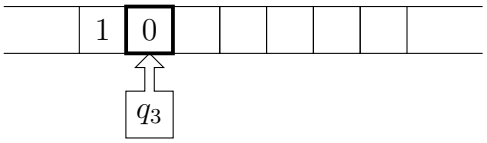
[†]Para fines prácticos respecto a la máquina universal, en algunas ocasiones denotaremos el estado inicial con q_1 y el estado final con q_{n+1}, donde n es el número de estados de la máquina que se está definiendo.



Estando sobre una celda vacía en el estado q_1, $\mathcal{M}$ imprime un 1, mueve el cabezal un lugar a la derecha y cambia al estado q_2.



Una vez que está sobre una celda vacía en el estado q_2, $\mathcal{M}$ imprime un 0, deja el cabezal en la misma posición y cambia al estado final q_3.



Claramente $\mathcal{M}(\varepsilon)\downarrow 10$.

Puesto que en algunas ocasiones trabajaremos con máquinas cuya función de transición es bastante grande, nos tomaremos la libertad de permitir que las máquinas impriman más de un símbolo o realicen más de un movimiento antes de cambiar de estado. Asimismo, omitiremos el uso de la letra N, entendiendo que si no se indica movimiento hacia la izquierda o la derecha, el cabezal debe permanecer en la misma posición. De igual forma omitiremos el uso del símbolo $\llcorner\!\lrcorner$, indicando únicamente el posible movimiento del cabezal, en el caso de la columna de operaciones, mientras que en el caso de la columna de símbolos, indicaremos que no hay ningún símbolo escrito sobre la cinta. Cuando no se indique que debe imprimirse algún símbolo, se debe entender que el símbolo escrito sobre la celda debe permanecer sin modificarse. Esto con la finalidad de reducir considerablemente el tamaño de las funciones de transición. A continuación mostramos un ejemplo.

Ejemplo 1.4. Consideremos la máquina que escribe la sucesión $= 0\llcorner\!\lrcorner 1\llcorner\!\lrcorner 0\llcorner\!\lrcorner 1\llcorner\!\lrcorner 0\llcorner\!\lrcorner 1\llcorner\!\lrcorner \ldots$ descrita por la siguiente tabla:

Estado	*Símbolo*	*Operaciones*	*Estado final*
q_1	$\sqcup$	$P0, R$	q_2
q_2	$\sqcup$	$P\sqcup, R$	q_3
q_3	$\sqcup$	$P1, R$	q_4
q_4	$\sqcup$	$P\sqcup, R$	q_1

Si adoptamos la convención antes mencionada, la tabla anterior puede reducirse como sigue:

Estado	*Símbolo*	*Operaciones*	*Estado final*
q_1	Ninguno	$P0$	q_1
	0	$R, R, P1$	q_1
	1	$R, R, P0$	q_1

Aunque en un principio parece que se ha alterado el comportamiento de la máquina, debemos recordar que no es así, simplemente se ha abreviado su tabla y puede devolverse a su forma original agregando los estados necesarios.

1.2.3. La máquina universal $\mathcal{U}$

Sea $\{T_i\}_{i\in\mathbb{N}}$ una enumeración de máquinas de Turing. Una máquina de Turing universal es cualquier máquina de Turing $\mathcal{U}$ tal que $\mathcal{U}(i, w) = T_i(w)$, para toda $i \in \mathbb{N}$ y para toda w en el alfabeto de T_i.

En esta sección incluiremos la descripción hecha por el propio Turing de una máquina de ese tipo en su famoso artículo ya mencionado [15] y la analizaremos mediante un ejemplo. Para ello, necesitaremos algunas herramientas previas que a continuación definimos y listamos.

1.2.3.1. Subrutinas

Dentro de la programación suele conocerse como *subrutina* a un conjunto de instrucciones que se invoca desde un programa para ejecutar una tarea específica y una vez que ha completado dicha tarea, pasa el "control" a otra subrutina, o bien al programa principal. Para poder realizar una descripción sencilla de la máquina $\mathcal{U}$ recurriremos a estas herramientas, pues de acuerdo con Turing:

> Hay cierto tipo de procesos usados por casi todas las máquinas, y éstos, en algunas máquinas, son usados en varias conexiones. Estos procesos incluyen copiar sucesiones de símbolos, comparar sucesiones, borrar todos los símbolos de una forma dada, etc. En lo que respecta a estos procesos, podemos abreviar las tablas para las m-configuraciones[1] considerablemente mediante el uso de "tablas esqueleto"[2]. En las tablas esqueleto aparecen letras alemanas mayúsculas y letras griegas minúsculas. Éstas son de la naturaleza de "variables". Al reemplazar cada letra alemana mayúscula por una m-configuración y cada letra griega minúscula por un símbolo, obtenemos la tabla para una m-configuración.
> Las tablas esqueleto no deben considerarse más que como abreviaturas: no son esenciales.[3]

A continuación presentamos una lista de las subrutinas que serán necesarias para describir a $\mathcal{U}$ y explicamos brevemente las diferentes tareas que realizan. Para dar mayor claridad, les asignamos un número.

Subrutina 1.

Estado	*Símbolo*	*Operaciones*	*Estado final*
$\mathfrak{f}(\mathfrak{C},\mathfrak{B},\alpha)$	ə[4]	L	$\mathfrak{f}_1(\mathfrak{C},\mathfrak{B},\alpha)$
	Distinto de ə	L	$\mathfrak{f}(\mathfrak{C},\mathfrak{B},\alpha)$
	Ninguno	L	$\mathfrak{f}(\mathfrak{C},\mathfrak{B},\alpha)$
$\mathfrak{f}_1(\mathfrak{C},\mathfrak{B},\alpha)$	α		$\mathfrak{C}$
	Distinto de α	R	$\mathfrak{f}_1(\mathfrak{C},\mathfrak{B},\alpha)$
	Ninguno	R	$\mathfrak{f}_2(\mathfrak{C},\mathfrak{B},\alpha)$
$\mathfrak{f}_2(\mathfrak{C},\mathfrak{B},\alpha)$	α		$\mathfrak{C}$
	Distinto de α	R	$\mathfrak{f}_1(\mathfrak{C},\mathfrak{B},\alpha)$
	Ninguno	R	$\mathfrak{B}$

[1] Turing utiliza el término "m-configuración" para referirse a lo que nosotros conocemos como "estado".

[2] Turing utiliza el término "tabla esqueleto" para referirse a lo que nosotros conocemos como "subrutina".

[3] Turing, [19], pp. 121-122

[4] Turing utiliza este símbolo para delimitar el inicio de la sucesión escrita sobre la cinta.

El funcionamiento de esta subrutina consiste en mover el cabezal hasta el inicio de la cinta para posteriormente buscar el primer "α". Una vez que lo encuentra se detiene sobre él y cambia al estado "$\mathfrak{C}$". Si no hay ningún "α" escrito, mueve el cabezal hasta el final de la sucesión escrita sobre la cinta y cambia al estado "$\mathfrak{B}$".

Subrutina 2.

Estado	*Símbolo*	*Operaciones*	*Estado final*
$\mathfrak{e}(\mathfrak{C},\mathfrak{B},\alpha)$			$\mathfrak{f}(\mathfrak{e}_1(\mathfrak{C},\mathfrak{B},\alpha),\mathfrak{B},\alpha)$
$\mathfrak{e}_1(\mathfrak{C},\mathfrak{B},\alpha)$		E	$\mathfrak{C}$

Puesto que para definir esta subrutina se recurre a la anterior, su tabla puede resultar confusa. Sin embargo, si hacemos las debidas sustituciones en la tabla de $\mathfrak{f}$, podemos darnos cuenta de que el funcionamiento de $\mathfrak{e}$ consiste en "encontrar" el primer "α" sobre la cinta, borrarlo en caso de que exista y cambiar al estado "$\mathfrak{C}$". En otro caso, simplemente mueve el cabezal hasta el final de la sucesión escrita sobre la cinta y cambia al estado "$\mathfrak{B}$".

Subrutina 3.

Estado	*Símbolo*	*Operaciones*	*Estado final*
$\mathfrak{e}(\mathfrak{B},\alpha)$			$\mathfrak{e}(\mathfrak{e}(\mathfrak{B},\alpha),\mathfrak{B},\alpha)$

Como podemos notar, esta subrutina y la anterior poseen el mismo nombre, por lo que únicamente pueden distinguirse mediante su número de argumentos[5]. Además, la versión de dos argumentos se define a partir de la versión de tres argumentos. Si realizamos las debidas sustituciones, podemos observar que el funcionamiento de $\mathfrak{e}(\mathfrak{B},\alpha)$ consiste en borrar todos los "α" escritos sobre la cinta, posteriormente mueve el cabezal al final de la cinta y cambia al estado $\mathfrak{B}$. Si no hay ningún "α", simplemente mueve el cabezal al final de la cinta y cambia al estado $\mathfrak{B}$.

[5]En el ámbito de la programación, esto se conoce como *sobrecarga de funciones*.

Subrutina 4.

Estado	*Símbolo*	*Operaciones*	*Estado final*
$\mathfrak{pe}(\mathfrak{C},\beta)$			$\mathfrak{f}(\mathfrak{pe}_1(\mathfrak{C},\beta),\mathfrak{C},\mathfrak{d})$
$\mathfrak{pe}_1(\mathfrak{C},\beta)$	Cualquiera	R,R	$\mathfrak{pe}_1(\mathfrak{C},\beta)$
	Ninguno	$P\beta$	$\mathfrak{C}$

Su funcionamiento consiste en buscar el primer "$\mathfrak{d}$" sobre la cinta (*i.e.* buscar el "inicio" de la cinta), mover el cabezal hacia la derecha hasta el final de la sucesión escrita, escribir el símbolo "β" y cambiar al estado $\mathfrak{C}$.

Subrutina 5.

Estado	*Símbolo*	*Operaciones*	*Estado final*
$\mathfrak{l}(\mathfrak{C})$		L	$\mathfrak{C}$

Consiste simplemente en mover el cabezal un lugar a la izquierda y cambiar al estado $\mathfrak{C}$.

Subrutina 6.

Estado	*Símbolo*	*Operaciones*	*Estado final*
$\mathfrak{r}(\mathfrak{C})$		R	$\mathfrak{C}$

Consiste simplemente en mover el cabezal un lugar a la derecha y cambiar al estado $\mathfrak{C}$.

Subrutina 7.

Estado	*Símbolo*	*Operaciones*	*Estado final*
$\mathfrak{f}'(\mathfrak{C},\mathfrak{B},\alpha)$			$\mathfrak{f}(\mathfrak{l}(\mathfrak{C}),\mathfrak{B},\alpha)$

El comportamiento de esta subrutina es casi igual al de $\mathfrak{f}$, salvo porque mueve el cabezal un lugar a la izquierda antes de cambiar al estado $\mathfrak{C}$.

Subrutina 8.

Estado	*Símbolo*	*Operaciones*	*Estado final*
$\mathfrak{f}''(\mathfrak{C}, \mathfrak{B}, \alpha)$			$\mathfrak{f}(\mathfrak{r}(\mathfrak{C}), \mathfrak{B}, \alpha)$

Su comportamiento es casi igual al de $\mathfrak{f}$, salvo porque mueve el cabezal un lugar a la derecha antes de cambiar al estado $\mathfrak{C}$.

Subrutina 9.

Estado	*Símbolo*	*Operaciones*	*Estado final*
$\mathfrak{c}(\mathfrak{C}, \mathfrak{B}, \alpha)$			$\mathfrak{f}'(\mathfrak{c}_1(\mathfrak{C}), \mathfrak{B}, \alpha)$
$\mathfrak{c}_1(\mathfrak{C})$	β		$\mathfrak{pe}(\mathfrak{C}, \beta)$

Lo que hace es copiar el símbolo marcado[6] con "α" al final de la sucesión escrita sobre la cinta y cambia al estado $\mathfrak{C}$. Si no hay ningún símbolo marcado con "α", mueve el cabezal al final de la sucesión y cambia al estado $\mathfrak{B}$. Nótese que $\mathfrak{c}_1$ no es exactamente una subrutina. Más bien, es un esquema que dará lugar a una subrutina específica para cada símbolo "β".

Subrutina 10.

Estado	*Símbolo*	*Operaciones*	*Estado final*
$\mathfrak{ce}(\mathfrak{C}, \mathfrak{B}, \alpha)$			$\mathfrak{c}(\mathfrak{e}(\mathfrak{C}, \mathfrak{B}, \alpha), \mathfrak{B}, \alpha)$

Es prácticamente idéntica a $\mathfrak{c}$, salvo porque borra el marcador "α" después de haber copiado el símbolo correspondiente y antes de cambiar al estado $\mathfrak{C}$.

[6] A diferencia de la convención actual, en la que escribimos todos los símbolos en celdas contiguas y "marcamos" un símbolo "x" con un símbolo "α" mediante una sustitución temporal, Turing considera dos tipos de celdas alternados sobre la cinta, a los que llama F-celdas y E-celdas. Las primeras se utilizan para escribir los símbolos que formarán parte del resultado del cómputo, mientras que las segundas se utilizan para colocar marcadores que serán borrados al finalizar el cómputo. De acuerdo con esto, Turing dice que "x" está marcado con "α" si "α" se encuentra a la derecha de "x".

Subrutina 11.

Estado	*Símbolo*	*Operaciones*	*Estado final*
$\mathfrak{ce}(\mathfrak{B},\alpha)$			$\mathfrak{ce}(\mathfrak{ce}(\mathfrak{B},\alpha),\mathfrak{B},\alpha)$

En esta subrutina nuevamente se recurre a la *sobrecarga de funciones*. Su comportamiento consiste en copiar al final de la sucesión escrita sobre la cinta todos los símbolos marcados con "α" en el mismo orden en que están escritos, borrando los respectivos marcadores, para posteriormente cambiar al estado $\mathfrak{B}$. Si no hay ningún símbolo marcado con "α", simplemente mueve el cabezal al final de la sucesión escrita y cambia al estado "$\mathfrak{B}$".

Subrutina 12.

Estado	*Símbolo*	*Operaciones*	*Estado final*
$\mathfrak{cp}(\mathfrak{C},\mathfrak{U},\mathfrak{E},\alpha,\beta)$			$\mathfrak{f}'(\mathfrak{cp}_1(\mathfrak{C},\mathfrak{U},\beta),\mathfrak{f}(\mathfrak{U},\mathfrak{E},\beta),\alpha)$
$\mathfrak{cp}_1(\mathfrak{C},\mathfrak{U},\beta)$	γ		$\mathfrak{f}'(\mathfrak{cp}_2(\mathfrak{C},\mathfrak{U},\gamma),\mathfrak{U},\beta)$
$\mathfrak{cp}_2(\mathfrak{C},\mathfrak{U},\gamma)$	γ		$\mathfrak{C}$
	Distinto de γ		$\mathfrak{U}$

Esta subrutina es mucho más complicada que las anteriores, pues está definida mediante esquemas que dan lugar a subrutinas específicas bajo ciertas condiciones. Su funcionamiento consiste en buscar el primer símbolo marcado con "α" y el primer símbolo marcado con "β" y compararlos. Si son iguales, cambia al estado $\mathfrak{C}$. Si no hay ningún símbolo marcado con "α" ni con "β", mueve el cabezal al final de la sucesión escrita y cambia al estado "$\mathfrak{E}$". Si no se da ninguno de los dos casos anteriores, mueve el cabezal al final de la cinta y cambia al estado "$\mathfrak{U}$".

Subrutina 13.

Estado	*Símbolo*	*Operaciones*	*Estado final*
$\mathfrak{cpe}(\mathfrak{C},\mathfrak{U},\mathfrak{E},\alpha,\beta)$			$\mathfrak{cp}(\mathfrak{e}(\mathfrak{e}(\mathfrak{C},\mathfrak{C},\beta),\mathfrak{C},\alpha),\mathfrak{U},\mathfrak{E},\alpha,\beta)$

Es prácticamente idéntica a $\mathfrak{cp}$, salvo porque borra el primer "α" y el primer "β" cuando ambos existen y sus símbolos marcados coinciden.

Subrutina 14.

Estado	*Símbolo*	*Operaciones*	*Estado final*
$\mathfrak{cpe}(\mathfrak{U},\mathfrak{E},\alpha,\beta)$			$\mathfrak{cpe}(\mathfrak{cpe}(\mathfrak{U},\mathfrak{E},\alpha,\beta),\mathfrak{U},\mathfrak{E},\alpha,\beta)$

Esta subrutina recurre nuevamente a una *sobrecarga de funciones*. Su funcionamiento consiste en comparar las sucesiones marcadas con "α" y "β" elemento por elemento. Si son iguales, borra todos los "α" y todos los "β" y cambia al estado $\mathfrak{E}$. En cualquier otro caso, cambia al estado $\mathfrak{U}$ y borra únicamente los marcadores de los primeros símbolos que coincidieron. Por ejemplo, si comparara las sucesiones $D\alpha A\alpha A\alpha A\alpha$ y $D\beta A\beta A\beta C\beta$, las dejaría marcadas como $DAAA\alpha$ y $DAAC\beta$. Por otro lado, si comparara las sucesiones $D\alpha A\alpha A\alpha A\alpha$ y $C\beta A\beta A\beta A\beta$, las dejaría intactas.

Subrutina 15.

Estado	*Símbolo*	*Operaciones*	*Estado final*
$\mathfrak{q}(\mathfrak{C})$	Cualquiera	R	$\mathfrak{q}(\mathfrak{C})$
	Ninguno	R	$\mathfrak{q}_1(\mathfrak{C})$
$\mathfrak{q}_1(\mathfrak{C})$	Cualquiera	R	$\mathfrak{q}(\mathfrak{C})$
	Ninguno		$\mathfrak{C}$

Simplemente mueve el cabezal hacia la derecha hasta encontrar dos celdas en blanco seguidas (*i.e.* hasta encontrar el final de la sucesión escrita) para posteriormente cambiar al estado $\mathfrak{C}$.

Subrutina 16.

Estado	*Símbolo*	*Operaciones*	*Estado final*
$\mathfrak{q}(\mathfrak{C},\alpha)$			$\mathfrak{q}(\mathfrak{q}_1(\mathfrak{C},\alpha))$
$\mathfrak{q}_1(\mathfrak{C},\alpha)$	α		$\mathfrak{C}$
	Distinto de α	L	$\mathfrak{q}_1(\mathfrak{C},\alpha)$

Esta definición nuevamente recurre a una *sobrecarga de funciones*. Su comportamiento consiste en mover el cabezal hasta el final de la sucesión escrita sobre la cinta, posteriormente moverlo hacia la izquierda hasta encontrar "α" y finalmente cambiar al estado $\mathfrak{C}$. Nótese que esta subrutina únicamente debe utilizarse cuando haya al menos un "α" escrito sobre la cinta, pues, en caso contrario, el cabezal se movería hacia la izquierda sin detenerse.

Subrutina 17.

Estado	*Símbolo*	*Operaciones*	*Estado final*
$\mathfrak{pe}_2(\mathfrak{C},\alpha,\beta)$			$\mathfrak{pe}(\mathfrak{pe}(\mathfrak{C},\beta),\alpha)$

Simplemente imprime $\alpha\beta$ al final de la sucesión escrita sobre la cinta y cambia al estado $\mathfrak{C}$.

Subrutina 18.

Estado	*Símbolo*	*Operaciones*	*Estado final*
$\mathfrak{ce}_2(\mathfrak{B},\alpha,\beta)$			$\mathfrak{ce}(\mathfrak{ce}(\mathfrak{B},\beta),\alpha)$

Esta subrutina primero copia al final de la sucesión escrita todos los símbolos marcados con "α" y luego todos los símbolos marcados con "β", borrando los respectivos marcadores, y cambia al estado $\mathfrak{B}$.

Subrutina 19.

Estado	*Símbolo*	*Operaciones*	*Estado final*
$\mathfrak{ce}_3(\mathfrak{B},\alpha,\beta,\gamma)$			$\mathfrak{ce}(\mathfrak{ce}_2(\mathfrak{B},\beta,\gamma),\alpha)$

Es la análoga a $\mathfrak{ce}_2$ con tres marcadores en vez de dos.

Subrutina 20.

Estado	*Símbolo*	*Operaciones*	*Estado final*
$\mathfrak{ce}_4(\mathfrak{B},\alpha,\beta,\gamma,\delta)$			$\mathfrak{ce}(\mathfrak{ce}_3(\mathfrak{B},\beta,\gamma,\delta),\alpha)$

Es la análoga a $\mathfrak{ce}_3$ con cuatro marcadores en vez de tres.

Subrutina 21.

Estado	*Símbolo*	*Operaciones*	*Estado final*
$\mathfrak{ce}_5(\mathfrak{B},\alpha,\beta,\gamma,\delta,\varepsilon)$			$\mathfrak{ce}(\mathfrak{ce}_4(\mathfrak{B},\beta,\gamma,\delta,\varepsilon),\alpha)$

Es la análoga a $\mathfrak{ce}_4$ con cinco marcadores en vez de cuatro.

Subrutina 22.

Estado	*Símbolo*	*Operaciones*	*Estado final*
$\mathfrak{e}(\mathfrak{C})$	ə	R	$\mathfrak{e}_1(\mathfrak{C})$
	Distinto de ə	L	$\mathfrak{e}(\mathfrak{C})$
$\mathfrak{e}_1(\mathfrak{C})$	Cualquiera	R,E,R	$\mathfrak{e}_1(\mathfrak{C})$
	Ninguno		$\mathfrak{C}$

Nuevamente tenemos un caso de *sobrecarga de funciones*. Sin embargo, a diferencia de los casos anteriores, esta definición no utiliza ninguna de sus homónimas previamente definidas. Su funcionamiento consiste en borrar de izquierda a derecha todos los marcadores escritos sobre la cinta y posteriormente cambiar al estado $\mathfrak{C}$.

Subrutina 23.

Estado	*Símbolo*	*Operaciones*	*Estado final*
$\mathfrak{con}(\mathfrak{C},\alpha)$	Distinto de A	R,R	$\mathfrak{con}(\mathfrak{C},\alpha)$
	A	$L,P\alpha,R$	$\mathfrak{con}_1(\mathfrak{C},\alpha)$
$\mathfrak{con}_1(\mathfrak{C},\alpha)$	A	$R,P\alpha,R$	$\mathfrak{con}_1(\mathfrak{C},\alpha)$
	D	$R,P\alpha,R$	$\mathfrak{con}_2(\mathfrak{C},\alpha)$
	Ninguno	$PD,R,P\alpha,R,R,R$	$\mathfrak{C}$
$\mathfrak{con}_2(\mathfrak{C},\alpha)$	C	$R,P\alpha,R$	$\mathfrak{con}_2(\mathfrak{C},\alpha)$
	Distinto de C	R,R	$\mathfrak{C}$

Se define para usarse exclusivamente en la máquina universal $\mathcal{U}$. Su funcionamiento consiste en marcar con "α" a alguna pareja (q_i, s_j) de la máquina "imitada" por $\mathcal{U}$ y posteriormente cambia al estado $\mathfrak{C}$.

Éstas son todas las subrutinas que necesitamos para describir a $\mathcal{U}$. Adicionalmente a esto, necesitaremos codificar las funciones de transición de la máquina a imitar mediante una sucesión de símbolos que a continuación explicamos.

1.2.3.2. Descripciones estándar y números de descripción

Para facilitar el funcionamiento de la máquina universal $\mathcal{U}$, codificaremos la función de transición de la máquina a imitar como sigue:

- Como primer paso, escribiremos la máquina como función de transición formal.
- Como segundo paso, escribiremos la función de transición en un solo renglón, separando cada asignación mediante un punto y coma. Por ejemplo, la máquina $\mathcal{M}$ del ejemplo 1.3 quedaría escrita como $q_1 \llcorner 1 R q_2$; $q_2 \llcorner 0 N q_3$;.
- Posteriormente, reescribiremos lo anterior mediante la siguiente codificación:
  - El estado "q_i" se codifica por medio de la letra "D" seguida de la letra "A" repetida i veces. Por ejemplo, el estado "q_2" se codifica como "DAA".
  - El símbolo "S_j" se codifica por medio de la letra "D" seguida de la letra "C" repetida j veces. Por ejemplo, el símbolo "S_1" se codifica como "DC".
  - Los movimientos de la cinta se codifican con las mismas letras utilizadas en la función de transición (R = moverse a la derecha, L = moverse a la izquierda, N = no moverse).

  De acuerdo con las convenciones anteriores, la máquina $\mathcal{M}$ queda codificada como $DADDCCRDAA; DAADDCNDAAA$. A este código se le conoce como *descripción estándar (D.E)* de $\mathcal{M}$, que será la utilizada por la máquina universal para imitar a $\mathcal{M}$.

Con el procedimiento anterior, podemos codificar cualquier máquina de Turing para imitarla mediante la máquina universal $\mathcal{U}$, que describimos más adelante. Adicionalmente a esto, podemos considerar la siguiente enumeración, por medio de la cual podemos codificar cualquier máquina de Turing mediante un número.

$$\begin{array}{ccccccc} A & C & D & L & R & N & ; \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array}$$

De acuerdo con esto, la máquina $\mathcal{M}$ del ejemplo 1.3 quedaría codificada por el número 31332253117311332263111, conocido como *número de descripción* de $\mathcal{M}$. Aunque Turing no usa estos números en la definición de su máquina universal, nosotros los utilizaremos en el capítulo 3 en la definición de complejidad Kolmogórov.

1.2.3.3. Descripción de $\mathcal{U}$

De acuerdo con Turing:

> Es posible inventar una sola máquina que puede ser utilizada para computar cualquier sucesión computable. Si a esta máquina se le suministra una cinta al inicio de la cual está escrita la $D.E$[7] de alguna máquina computadora[8] $\mathcal{M}$, entonces $\mathcal{U}$ computará la misma sucesión que $\mathcal{M}$.[9]

A continuación daremos la descripción[10] de $\mathcal{U}$ y analizaremos su funcionamiento mediante un ejemplo.

[7] $D.E.$ es la abreviatura de *descripción estándar* (véase la sección 1.2.3.2).

[8] Originalmente Turing llamó así a sus máquinas. El término "máquina de Turing" fue usado por primera vez en 1937 por Alonso Church.

[9] Turing, [19], pp. 127-128

[10] Incluimos la correcciones hechas por Post en [16].

Estado	*Símbolo*	*Operaciones*	*Estado final*
$\mathfrak{b}$			$\mathfrak{f}(\mathfrak{b}_1, \mathfrak{b}_1, ::)$
$\mathfrak{b}_1$		$R, R, P:, R, R, PD, R, R, PA$	$\mathfrak{anf}$
$\mathfrak{anf}$			$\mathfrak{q}(\mathfrak{anf}_1, :)$
$\mathfrak{anf}_1$			$\mathfrak{con}(\mathfrak{kom}, y)$
$\mathfrak{kom}$	;	R, Pz, L	$\mathfrak{con}(\mathfrak{kmp}, x)$
	z	L, L	$\mathfrak{kom}$
	Ni ; ni z	L	$\mathfrak{kom}$
$\mathfrak{kmp}$			$\mathfrak{cpe}(\mathfrak{e}(\mathfrak{e}(\mathfrak{anf}, x), y), \mathfrak{sim}, x, y)$
$\mathfrak{sim}$			$\mathfrak{f}'(\mathfrak{sim}_1, \mathfrak{sim}_2, z)$
$\mathfrak{sim}_1$			$\mathfrak{con}(\mathfrak{sim}_2, \_)$
$\mathfrak{sim}_2$	A		$\mathfrak{sim}_3$
	Distinto de A	L, Pu, R, R, R	$\mathfrak{sim}_2$
$\mathfrak{sim}_3$	Distinto de A	L, Py	$\mathfrak{e}(\mathfrak{mk}, z)$
	A	L, Py, R, R, R	$\mathfrak{sim}_3$
$\mathfrak{mk}$			$\mathfrak{q}(\mathfrak{mk}_1, :)$
$\mathfrak{mk}_1$	Distinto de A	R, R	$\mathfrak{mk}_1$
	A	L, L, L, L	$\mathfrak{mk}_2$
$\mathfrak{mk}_2$	C	R, Px, L, L, L	$\mathfrak{mk}_2$
	:		$\mathfrak{mk}_4$
	D	R, Px, L, L, L	$\mathfrak{mk}_3$

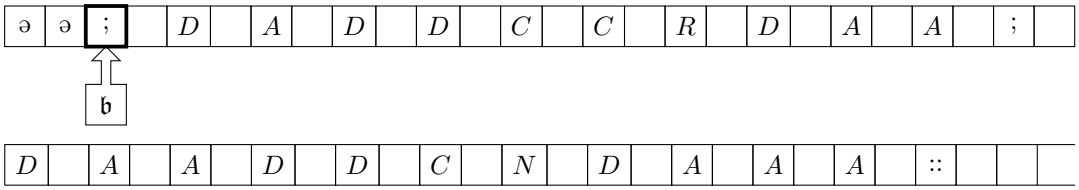
Estado	*Símbolo*	*Operaciones*	*Estado final*
$\mathfrak{mk}_3$	Distinto de : :	R, Pv, L, L, L	$\mathfrak{mk}_3$ $\mathfrak{mk}_4$
$\mathfrak{mk}_4$			$\mathfrak{con}(\mathfrak{l}(\mathfrak{l}(\mathfrak{mk}_5)), \_)$
$\mathfrak{mk}_5$	Cualquiera Ninguno	R, Pw, R $P:$	$\mathfrak{mk}_5$ $\mathfrak{sh}$
$\mathfrak{sh}$			$\mathfrak{f}(\mathfrak{sh}_1, \mathfrak{inst}, u)$
$\mathfrak{sh}_1$		L, L, L	$\mathfrak{sh}_2$
$\mathfrak{sh}_2$	D Distinto de D	R, R, R, R	$\mathfrak{sh}_3$ $\mathfrak{inst}$
$\mathfrak{sh}_3$	C Distinto de C	R, R	$\mathfrak{sh}_4$ $\mathfrak{inst}$

Estado	*Símbolo*	*Operaciones*	*Estado final*
$\mathfrak{sh}_4$	C Distinto de C	R, R	$\mathfrak{sh}_5$ $\mathfrak{pe}_2(\mathfrak{inst}, 0, :)$
$\mathfrak{sh}_5$	C Distinto de C		$\mathfrak{inst}$ $\mathfrak{pe}_2(\mathfrak{inst}, 1, :)$
$\mathfrak{inst}$			$\mathfrak{q}(\mathfrak{l}(\mathfrak{inst}_1), u)$
$\mathfrak{inst}_1$	L R N	R, E R, E R, E	$\mathfrak{ce}_5(\mathfrak{ov}, v, y, x, u, w)$ $\mathfrak{ce}_5(\mathfrak{ov}, v, x, u, y, w)$ $\mathfrak{ce}_5(\mathfrak{ov}, v, x, y, u, w)$
$\mathfrak{ov}$			$\mathfrak{e}(\mathfrak{anf})$

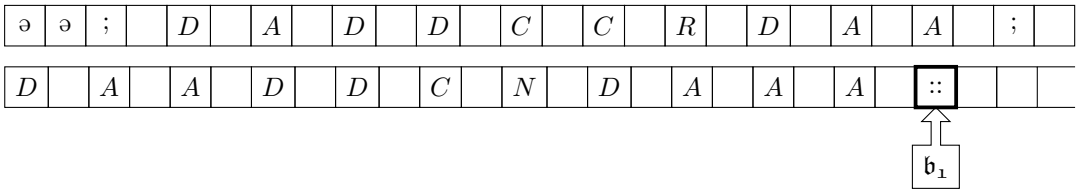
Debido a la gran cantidad de subrutinas utilizadas para definir a $\mathcal{U}$, su tabla puede resultar difícil de entender. A continuación analizamos un ca-

so particular de su comportamiento mediante un ejemplo y posteriormente hacemos algunas aclaraciones finales.

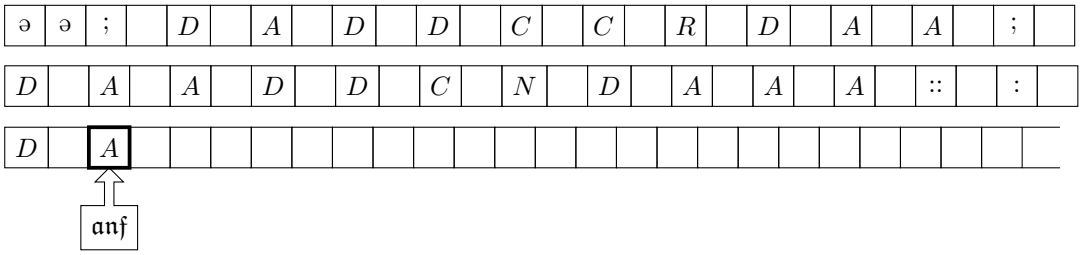
Ejemplo 1.5. Consideremos la máquina del ejemplo 1.3, cuya descripción estándar es $DADDCCRDAA;DAADDCNDAAA$.



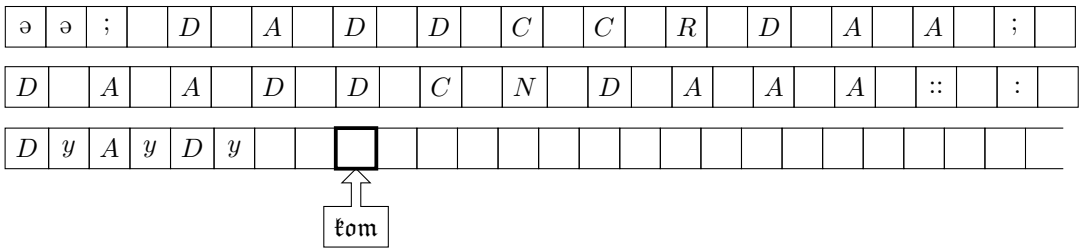
Como puede observarse, al iniciar $\mathcal{U}$, la descripción estándar de la máquina a emular debe estar escrita sobre F-celdas de la cinta, delimitada a la izquierda por dos marcadores "ə" y a la derecha por un marcador "::". La unidad de control debe encontrarse en el estado inicial $\mathfrak{b}$ y el cabezal debe encontrarse sobre el primer punto y coma.



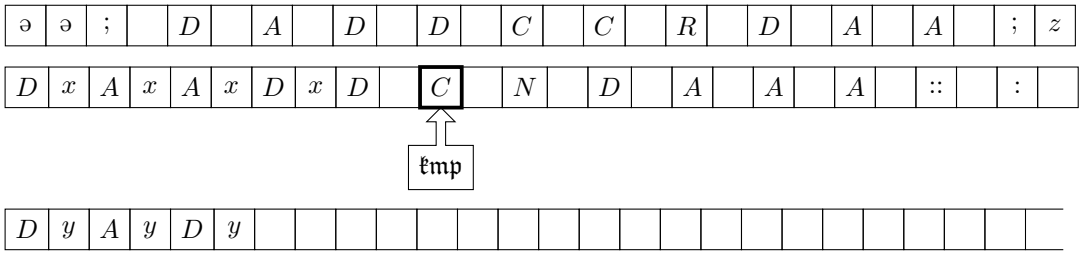
Una vez que está sobre cualquier símbolo en el estado $\mathfrak{b}$, $\mathcal{U}$ busca el último "::" escrito sobre la cinta, coloca el cabezal sobre él y cambia al estado $\mathfrak{b}_1$.



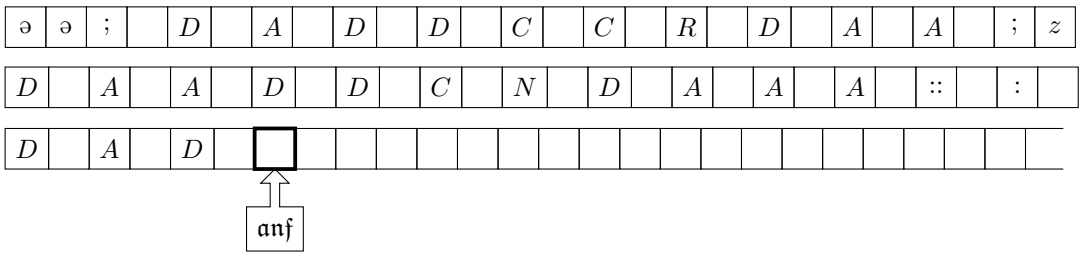
Cuando se encuentra en el estado $\mathfrak{b}_1$ sobre cualquier símbolo, $\mathcal{U}$ imprime la sucesión $:DA$ en las F-celdas siguientes y cambia al estado $\mathfrak{anf}$.



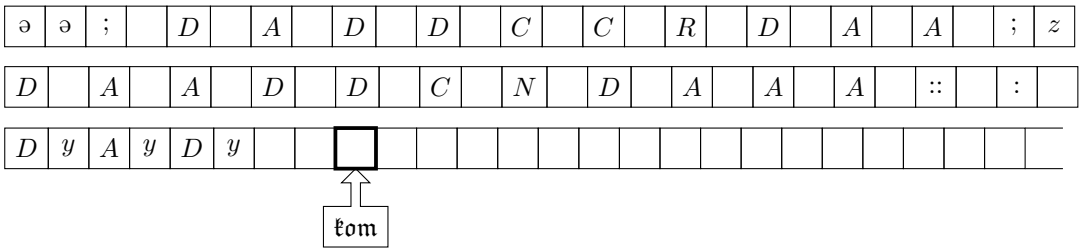
Cuando está sobre A en el estado $\mathfrak{anf}$, $\mathcal{U}$ marca con y el estado de $\mathcal{M}$ a imitar, mueve el cabezal dos celdas a la derecha y cambia al estado $\mathfrak{kom}$. Como en este caso solamente estaba indicado el estado q_1 sin ningún símbolo, $\mathcal{U}$ escribe una D en la F-celda posterior a A para indicarse a sí misma que la primera combinación a imitar es $(q_1, \llcorner\!\lrcorner)$.



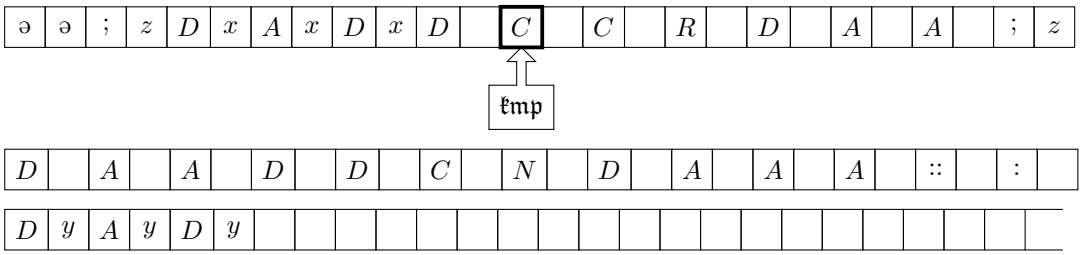
Estando sobre una celda vacía en el estado $\mathfrak{kom}$, $\mathcal{U}$ mueve el cabezal a la izquierda hasta encontrar un punto y coma sin marcar. Cuando lo encuentra lo marca con z, marca la pareja de estado y símbolo de $\mathcal{M}$ que está a su derecha con x, mueve el cabezal dos celdas a la derecha y cambia al estado $\mathfrak{kmp}$.



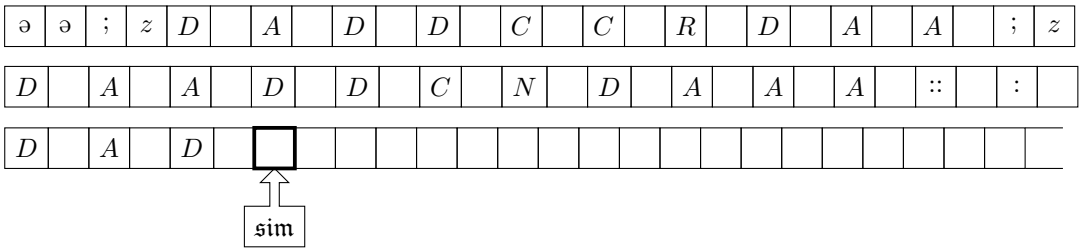
Ya que está sobre C en el estado $\mathfrak{kmp}$, $\mathcal{U}$ compara las sucesiones marcadas con x e y. Como en este caso son distintas, borra todos los marcadores x e y restantes y cambia al estado $\mathfrak{anf}$.



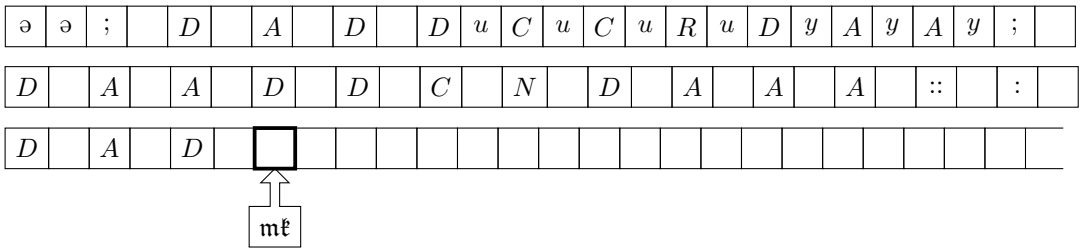
Estando sobre una celda vacía en el estado $\mathfrak{anf}$, $\mathcal{U}$ busca los últimos dos puntos escritos sobre la cinta, marca la pareja de estado y símbolo de $\mathcal{M}$ escrita a la derecha con y, mueve el cabezal dos celdas a la derecha y cambia al estado $\mathfrak{com}$.



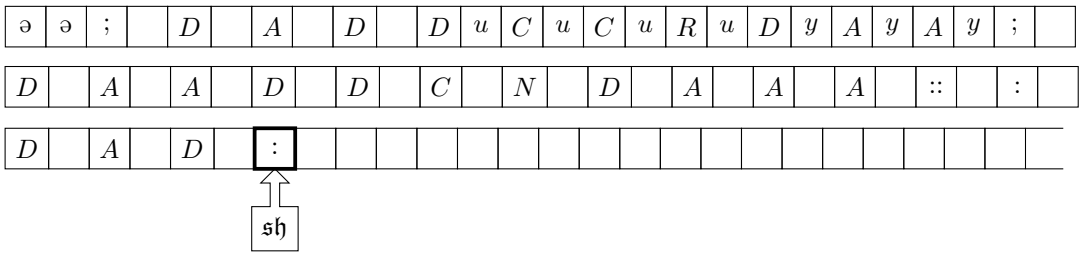
Una vez que está sobre una celda vacía en el estado $\mathfrak{com}$, $\mathcal{U}$ mueve el cabezal a la izquierda hasta encontrar un punto y coma sin marcar. Cuando lo encuentra lo marca con z, marca la pareja de estado y símbolo de $\mathcal{M}$ que está a su derecha con x, mueve el cabezal dos celdas a la derecha y cambia al estado $\mathfrak{cmp}$.



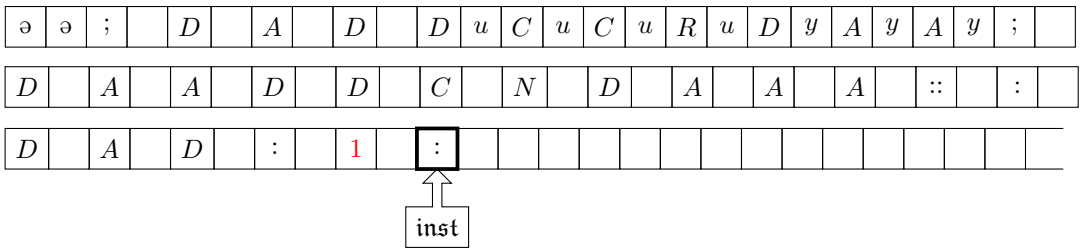
Estando sobre C en el estado $\mathfrak{cmp}$, $\mathcal{U}$ compara las sucesiones marcadas con x e y. Como en este caso coinciden, cambia al estado $\mathfrak{sim}$



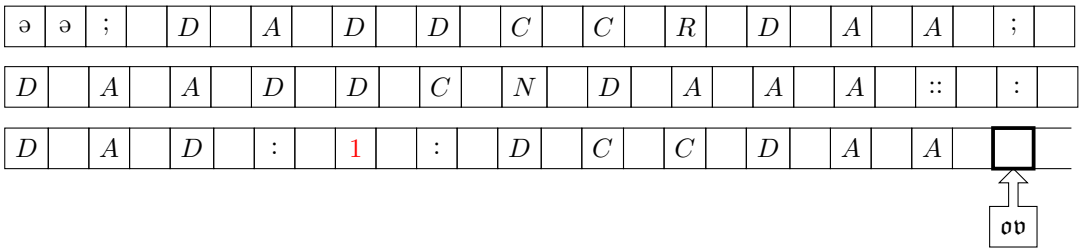
Ya que está sobre una celda vacía en el estado $\mathfrak{sim}$, $\mathcal{U}$ busca el primer z de izquierda a derecha, "salta" la pareja (q_i, s_j) que está a su derecha, marca las operaciones (símbolo a imprimir, movimiento) correspondientes a esa pareja con u y el estado final correspondiente con y; borra todos los marcadores z y cambia al estado $\mathfrak{mk}$.



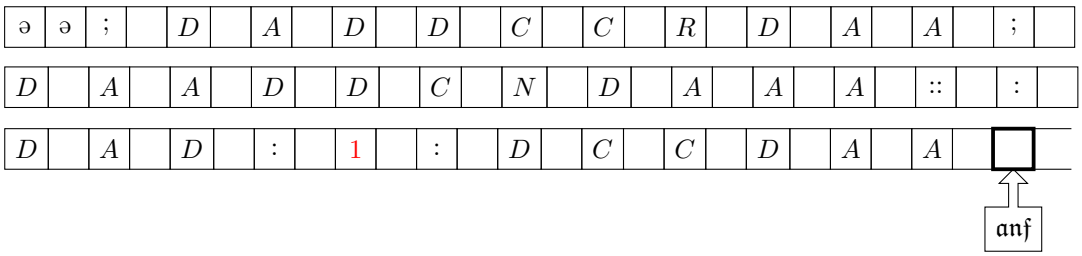
Estando sobre una celda vacía en el estado $\mathfrak{mk}$, $\mathcal{U}$ imprime : en la primera F-celda a la derecha del final de la sucesión escrita sobre la cinta y cambia al estado $\mathfrak{sh}$.



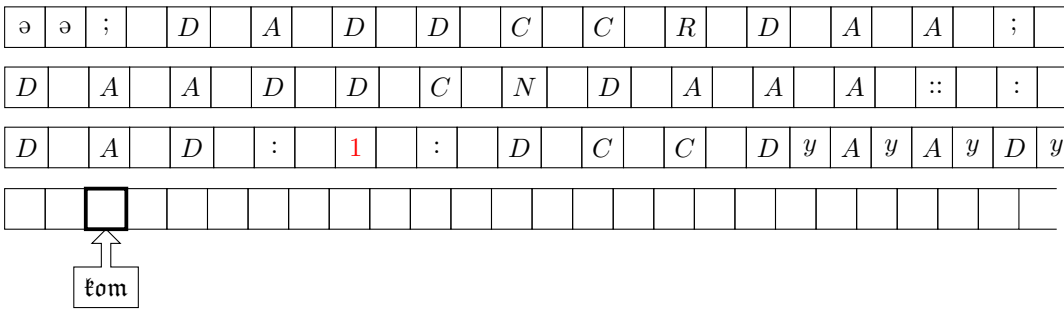
Cuando se encuentra sobre dos puntos en el estado $\mathfrak{sh}$, $\mathcal{U}$ verifica si las operaciones marcadas con u involucran imprimir 0 ó imprimir 1. Como en este caso involucran imprimir 1, $\mathcal{U}$ imprime 1 : en las F-celdas a la derecha del final de la sucesión y cambia al estado $\mathfrak{inst}$.



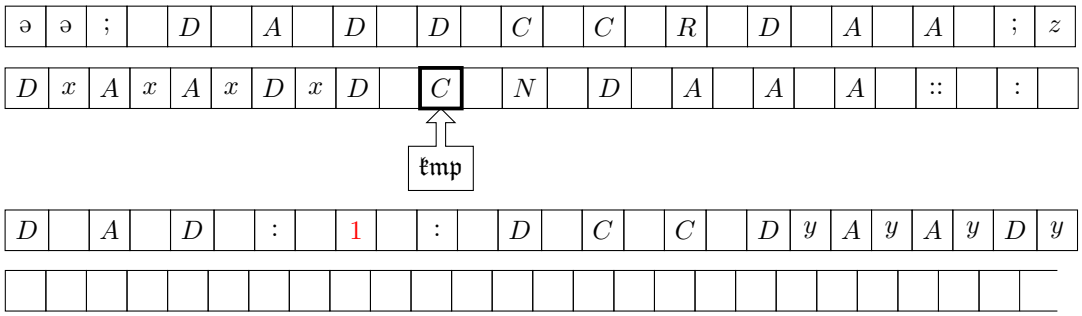
Una vez que está sobre dos puntos en el estado $\mathfrak{inst}$, $\mathcal{U}$ busca el último u sobre la cinta, verifica el símbolo que está marcando, en este caso R, borra este marcador, copia al "final" de la cinta la sucesión marcada con u y luego la sucesión marcada con y. Esto con la finalidad de indicarse a sí misma qué combinación estado y símbolo de $\mathcal{M}$ deberá imitar a continuación. Finalmente, cambia al estado $\mathfrak{ov}$.



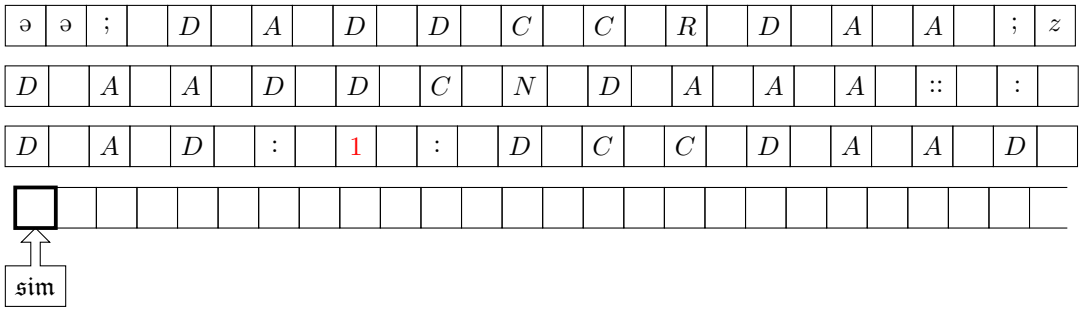
Estando sobre una celda vacía en el estado $\mathfrak{ov}$, $\mathcal{U}$ borra cualquier marcador que haya quedado sobre la cinta y cambia al estado $\mathfrak{anf}$.



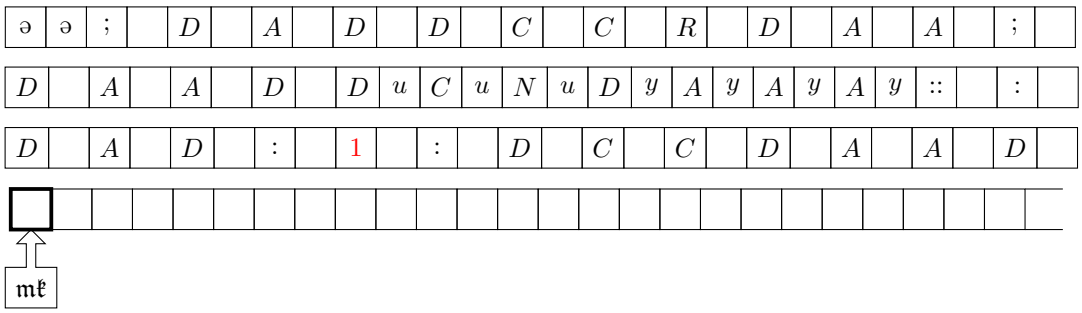
Cuando se encuentra sobre una celda vacía en el estado $\mathfrak{anf}$, $\mathcal{U}$ marca la próxima combinación de estado y símbolo a imitar, en este caso $(q_2, \sqcup)$ con y, mueve el cabezal tres celdas a la derecha y cambia al estado $\mathfrak{kom}$.



Estando sobre una celda vacía en el estado $\mathfrak{kom}$, $\mathcal{U}$ mueve el cabezal hacia la izquierda hasta encontrar un punto y coma, lo marca con z, marca con x la pareja de estado y símbolo que está a la derecha, mueve el cabezal tres celdas a la derecha y cambia al estado $\mathfrak{kmp}$.

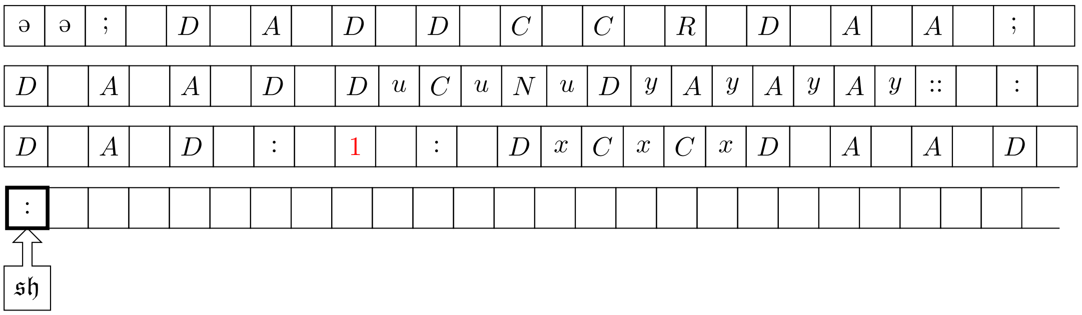


Ya que está sobre C en el estado $\mathfrak{kmp}$, $\mathcal{U}$ compara las sucesiones marcadas con x e y. Como en este caso son iguales, cambia al estado $\mathfrak{sim}$.

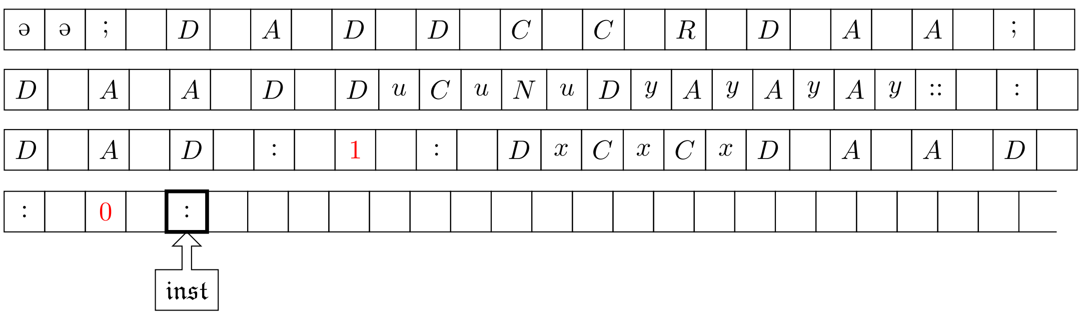


Estando sobre una celda vacía en el estado $\mathfrak{sim}$, $\mathcal{U}$ busca el primer z de izquierda a derecha, "salta" la pareja (q_i, s_j) ubicada a su derecha, marca las operaciones (símbolo a imprimir, movimiento) correspondientes a esa pareja

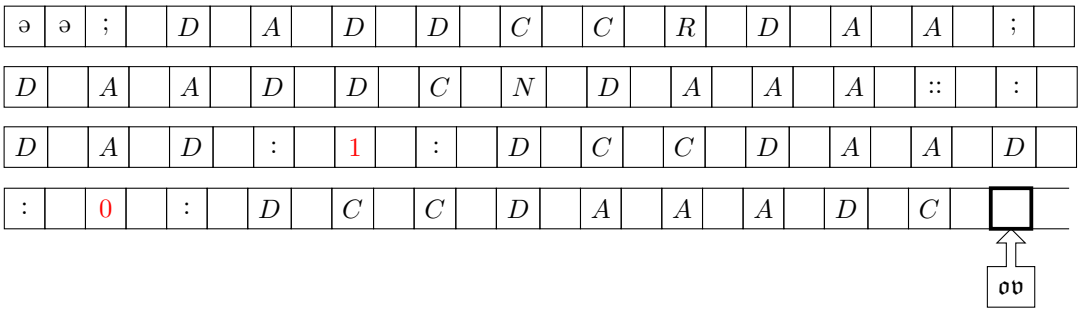
con u y el estado final correspondiente con y; borra todos los marcadores z y cambia al estado $\mathfrak{mk}$.



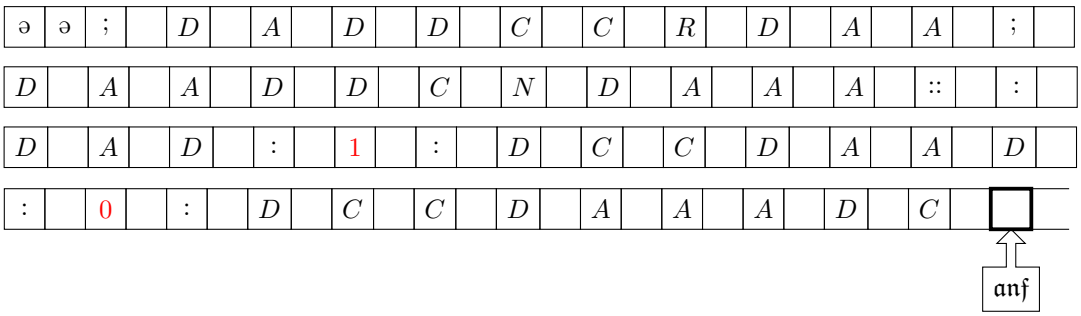
Una vez que está sobre una celda vacía en el estado $\mathfrak{mk}$, $\mathcal{U}$ marca con x el último símbolo “escrito” por $\mathcal{M}$, brinca el estado escrito a su derecha, imprime : en la primera F-celda a su derecha y cambia al estado $\mathfrak{sh}$.



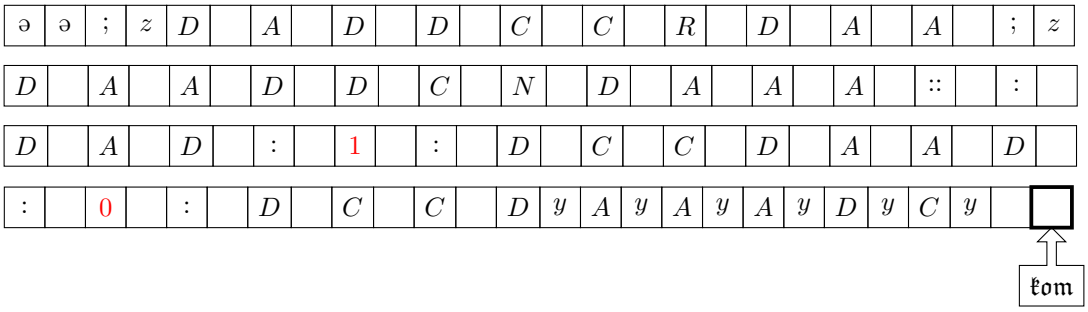
Estando sobre dos puntos en el estado $\mathfrak{sh}$, $\mathcal{U}$ verifica si las operaciones marcadas con u involucran imprimir 0 ó imprimir 1. Como en este caso involucran imprimir 0, $\mathcal{U}$ imprime 0 : en las F-celdas a la derecha del final de la sucesión y cambia al estado $\mathfrak{inst}$.



Una vez que está sobre dos puntos en el estado $\mathfrak{inst}$, $\mathcal{U}$ busca el último u sobre la cinta, verifica el símbolo que está marcando, en este caso N, borra este marcador, copia al "final" de la cinta la sucesión marcada con x, luego la sucesión marcada con y y luego la sucesión marcada con u. Esto con la finalidad de indicarse a sí misma qué combinación estado y símbolo de $\mathcal{M}$ deberá imitar a continuación. Finalmente, cambia al estado $\mathfrak{ov}$.



Estando sobre una celda vacía en el estado $\mathfrak{ov}$, $\mathcal{U}$ borra cualquier marcador que haya quedado sobre la cinta y cambia al estado $\mathfrak{anf}$.

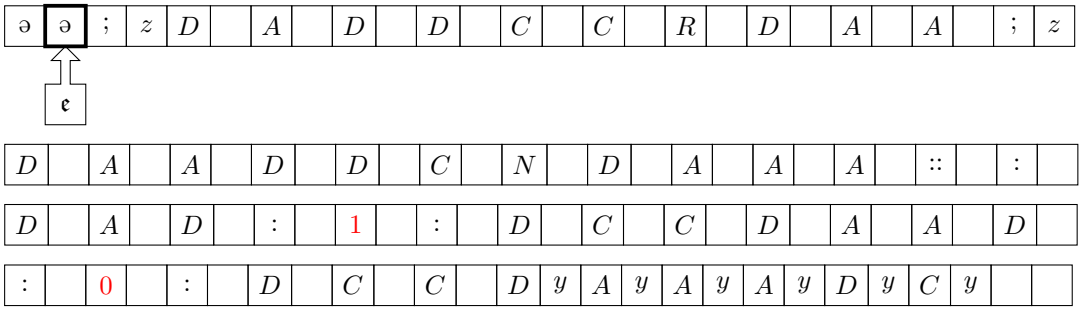


Ahora, $\mathcal{U}$ nuevamente repite el proceso de marcar la próxima combinación de estado y símbolo a imitar, en este caso $(q_3, 0)$. Sin embargo, como q_3 es

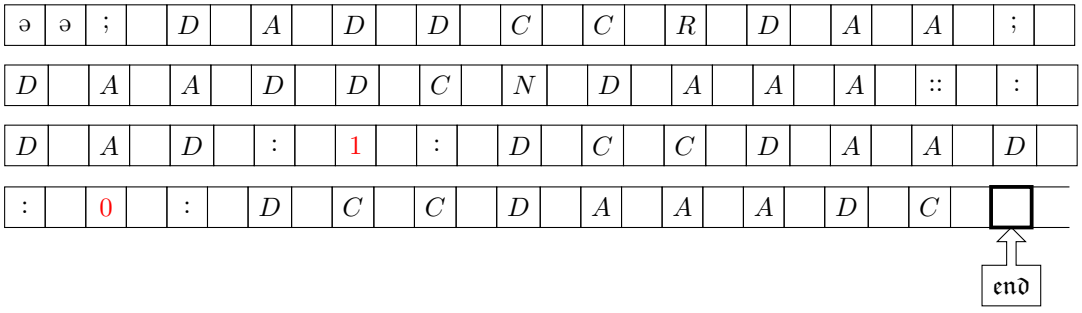
el estado final de $\mathcal{M}$, en algún punto todos los posibles estados a imitar estarán marcados con z, por lo que $\mathcal{U}$ continuará moviendo su cabezal hacia la izquierda infinitamente. Tal situación puede corregirse modificando la definición de $\mathfrak{tom}$ como sigue:

Estado	*Símbolo*	*Operaciones*	*Estado final*
$\mathfrak{tom}$	;	R, Pz, L	$\mathfrak{con}(\mathfrak{tmp},x)$
	z	L, L	$\mathfrak{tom}$
	ə		$\mathfrak{e}(\mathfrak{end})$
	Ni ; ni z ni ə	L	$\mathfrak{tom}$

Donde $\mathfrak{end}$ denota al estado final de $\mathcal{U}$. Con esta modificación el comportamiento de $\mathcal{U}$ sería el siguiente:



Al colocarse sobre "ə" en el estado $\mathfrak{tom}$, $\mathcal{U}$ borrará todos los marcadores restantes sobre la cinta y cambiará al estado final $\mathfrak{end}$ dando fin al cómputo. Notemos que esto sucederá únicamente cuando $\mathcal{U}$ pretenda imitar un estado no definido de $\mathcal{M}$. Es decir, cuando pretenda imitar un estado final.



Como conclusión, nos parece importante recalcar que:

I) El uso de subrutinas en la descripción de $\mathcal{U}$ claramente nos permite reducir su tamaño considerablemente.

II) La modificación realizada a $\mathfrak{tom}$ es necesaria, debido a que Turing definió su máquina universal con la intención de utilizarla para imitar únicamente cierto tipo de máquinas que él llamó *no circulares*, que son aquellas máquinas capaces de imprimir una infinidad de ceros y unos mediante un algoritmo finito y sin detenerse nunca.

III) $\mathcal{U}$ únicamente es capaz de imitar máquinas que escriban solamente ceros y unos, pues no cuenta con una instrucción apta para imprimir ningún otro símbolo.

IV) $\mathcal{U}$ solamente es capaz de imitar máquinas que tomen como dato de entrada a la palabra vacía ε. Si a $\mathcal{U}$ se le proporciona la *descripción estándar de cualquier* otro tipo de máquina, o un dato de entrada distinto de ε, tendrá un comportamiento errático.

V) Aunque podría decirse que $\mathcal{U}$ no imita fielmente a ninguna máquina $\mathcal{M}$, debemos notar que se debe a la necesidad de llevar un registro del cómputo.

VI) Si a $\mathcal{U}$ se le da la *descripción estándar* de una máquina $\mathcal{M}$ cuyo comportamiento involucre mover el cabezal a la izquierda, la sucesión de ceros y unos computada por $\mathcal{U}$ diferirá de la computada por $\mathcal{M}$ en el orden de uno o más elementos.

VII) Pese a todas estas limitaciones, $\mathcal{U}$ tiene una gran importancia histórica, pues su creación demostró que una sola máquina programable es suficiente para realizar cualquier cómputo posible. Hecho que más adelante fue retomado en el diseño de las computadoras.

En este capítulo explicamos brevemente el concepto de función recursiva. También definimos y explicamos ampliamente los conceptos de máquina de Turing y máquina de Turing universal, porque tienen un papel esencial en las definiciones del capítulo 3 y, consecuentemente, en varias demostraciones de los capítulos 4 y 5. En el siguiente capítulo definiremos y compararemos los conceptos de consistencia, ω-consistencia y completud.

Capítulo 2

Consistencia, ω-consistencia y correctud

Algunos de los principales conceptos utilizados en la demostración de los teoremas de Gödel son el de *consistencia*, el de *ω-consistencia* y el de *correctud*. A continuación introducimos un fragmento de la teoría requerida para definir estos conceptos (particularmente el de *correctud*). Posteriormente definimos y comparamos los conceptos antes mencionados.

2.1. Modelos de la aritmética

En esta sección analizamos el modelo *estándar* de la aritmética, mostramos la existencia de modelos *no estándar* y examinamos brevemente su estructura.

Cuando hablamos de la aritmética ($AR\backslash AP$), nos referimos a la teoría cuyo lenguaje cuenta con un predicado de igualdad, $t = s$, una constante individual 0, tres funciones s, $+$, $\cdot$, los axiomas propios

$P1)$ $\neg(sx = 0)$,

$P2)$ $sx_1 = sx_2 \rightarrow x_1 = x_2$,

$P3)$ $x_1 + 0 = x_1$,

$P4)$ $x_1 + sx_2 = s(x_1 + x_2)$,

$P5)$ $x_1 \cdot 0 = 0$,

$P6)$ $x_1 \cdot sx_2 = (x_1 \cdot x_2) + x_1$

y las reglas de inferencia

$$(M.P.)\frac{A, A \to B}{B}, \qquad (Gen.)\frac{A}{\forall nA}, \qquad (P.I.)\frac{A(0), A(n) \to A(sn)}{\forall nA(n)}.$$

Antes de comenzar a analizar los modelos de esta teoría, introducimos algunos conceptos y mostramos algunos resultados previos necesarios.

2.1.1. Algunos conceptos, nociones y resultados sobre modelos y números ordinales

Para poder definir claramente los conceptos de *completud* y *correctud*, es necesario utilizar algunos conceptos y resultados propios de la teoría de modelos, que a continuación introducimos.

2.1.1.1. Algunos conceptos de teoría de modelos

Definición 2.1. Sea L un lenguaje de primer orden de tipo τ y $\mathbf{M}$, $\mathbf{M}^*$ dos estructuras tipo τ de L con dominios D_M y D_{M^*} respectivamente. Decimos que $\mathbf{M}$ y $\mathbf{M}^*$ son *isomorfas* entre sí, si y sólo si existe una función biyectiva $g : D_M \to D_{M^*}$ que cumple lo siguiente:

1. Para cualquier predicado A_j^n de L y para cualesquiera $b_1, \ldots, b_n$ en D_M, $M \vDash A_j^n[b_1, \ldots, b_n]$ si y sólo si $M^* \vDash A_j^n[g(b_1), \ldots, g(b_n)]$.

2. Para cualquier función f_j^n de L y para cualesquiera $b_1, \ldots, b_n$ en D_M, $g((f_j^n)^M(b_1, \ldots, b_n)) = (f_n^j)^{M^*}(g(b_1), \ldots, g(b_n))$.

3. Para cualquier constante individual a_j de L, $g((a_j)^M) = (a_j)^{M^*}$.

Definición 2.2. Sean M_1 y M_2 dos estructuras de tipo τ correspondientes a un lenguaje de primer orden L. Decimos que M_1 es *elementalmente equivalente* a M_2 ($M_1 \equiv M_2$) si y sólo si los enunciados de L que son verdaderos en M_1 son los mismos enunciados de L que son verdaderos en M_2.

Definición 2.3. Sean M_1 y M_2 dos estructuras de tipo τ correspondientes a un lenguaje L, y sean D_1 y D_2 sus dominios. Decimos que M_2 es una *extensión* de M_1 ($M_1 \subseteq M_2$) si se cumple lo siguiente:

1. $D_1 \subseteq D_2$.

2. Para cualquier constante individual a_j de L, $(a_j)^{M_1} = (a_j)^{M_2}$.

3. Para cualquier función f_j^n de L y cualesquiera $b_1, \ldots, b_n$ en D_1, $(f_j^n)^{M_2}(b_1, \ldots, b_n) = (f_j^n)^{M_1}(b_1, \ldots, b_n)$.

4. Para cualquier predicado A_j^n de L y para cualesquiera $b_1, \ldots, b_n$ en D_1, $M_1 \vDash A_j^n[b_1, \ldots, b_n]$ si y sólo si $M_2 \vDash A_j^n[b_1, \ldots, b_n]$.

Definición 2.4. Sean M_1 y M_2 dos estructuras de tipo τ correspondientes a un lenguaje L. Decimos que M_2 es una *extensión elemental* de M_1 ($M_1 \leqslant_e M_2$) si

1. $M_1 \subseteq M_2$ y

2. para cualquier fórmula $\mathcal{B}(y_1, \ldots, y_n)$ de L y para cualesquiera $b_1, \ldots, b_n$ en el dominio de M_1, $M_1 \vDash \mathcal{B}[b_1, \ldots, b_n]$ si y sólo si $M_2 \vDash \mathcal{B}[b_1, \ldots, b_n]$ (*i.e.* $M_1 \equiv M_2$).

Definición 2.5. Sean M_1 y M_2 dos estructuras de tipo τ correspondientes a lenguaje L con dominios D_1 y D_2 respectivamente y $f : D_1 \to D_2$ una función. Decimos que f es un *encaje elemental* de M_1 en M_2 si y sólo si para cualquier fórmula $\mathcal{B}(x_1, \ldots, x_n)$ de L y para cualesquiera $a_1, \ldots, a_n$ en D_1 se cumple que $M_1 \vDash \mathcal{B}[a_1, \ldots, a_n]$ si y sólo si $M_2 \vDash \mathcal{B}[f(a_1), \ldots, f(a_n)]$. Es decir, si y sólo si f es un isomorfismo que va de M_1 a una subestructura elemental de M_2.

Definición 2.6. Sean T una teoría de primer orden con lenguaje L_T y M un modelo de T. Decimos que M es un *modelo primo* de T si admite un encaje elemental en cualquier modelo de T.

Teorema 2.1 (de compacidad)**.** Toda teoría axiomática T de primer orden es satisfacible si y sólo si es finitamente satisfacible.

Demostración. $\boxed{\Rightarrow}$ Como T es satisfacible, tiene una interpretación que la satisface y en particular satisface a todos sus subconjuntos finitos.

$\boxed{\Leftarrow}$ Supongamos que T no es satisfacible. Entonces, se tiene que $T \models \phi \wedge \neg\phi$ para cualquier fórmula ϕ. Por el teorema de completud de Gödel, esto implica que $\vdash_T \phi \wedge \neg\phi$. Como cualquier deducción en el cálculo de predicados es finita, existe Δ subconjunto finito de T tal que $\Delta \vdash_T \phi \wedge \neg\phi$. Luego $\Delta \models \phi \wedge \neg\phi$, por lo que Δ no es satisfacible. Por contraposición, se tiene que si Δ es satisfacible, T es satisfacible. □

2.1.1.2. Algunas nociones relativas a los números ordinales

Para poder definir los conceptos de *modelo estándar* y *modelo no estándar* de la aritmética, así como para hacer notar algunas propiedades y relaciones entre este tipo de modelos, son necesarias algunas nociones relativas a los números ordinales, que a continuación introducimos.

Definición 2.7. Sea X un conjunto. Decimos que X es *transitivo* si para todo $y \in X$, se cumple que $y \subseteq X$.

Definición 2.8. Un conjunto X es *un ordinal* si y sólo si es transitivo y está bien ordenado por la pertenencia restringida a X ($\in_X$).

Definición 2.9. Si $(A, \preceq)$ es un conjunto bien ordenado, su tipo de orden ($\|(A, \preceq)\|$) es el único ordinal isomorfo a A.

Definición 2.10. a) Dos funciones f y g son compatibles si para todo $x \in Dom(f) \cap Dom(g)$, $f(x) = g(x)$.

b) Un conjunto de funciones $\mathcal{F}$ es un sistema compatible de funciones si cualesquiera $f, g \in \mathcal{F}$ son compatibles.

Lema 2.1. Si f y g son compatibles, entonces $f \cup g$ es una función.

Demostración. Sean $(x, y), (x, z) \in f \cup g$. Entonces, $x \in Dom(f) \cup Dom(g)$.

Si $(x, y) \in Dom(f) \triangle Dom(g)$, necesariamente (x, y), $(x, z) \in f \backslash g$ ó $(x, y), (x, z) \in g \backslash f$, por lo que $y = z$.

Si $(x, y) \in Dom(f) \cap Dom(g)$, $y = f(x) = g(x) = z$ por hipótesis.

Por lo tanto, $f \cup g$ es una función. □

Teorema 2.2. Si $\mathcal{F}$ es un sistema compatible de funciones, entonces $\bigcup \mathcal{F}$ es una función con $Dom(\bigcup \mathcal{F}) = \bigcup\{Dom(f) | f \in \mathcal{F}\}$.

Demostración. Sean $(a,b),(a,c) \in \bigcup \mathcal{F}$. Entonces, existen $f_1, f_2 \in \mathcal{F}$ tales que $(a,b) \in f_1$ y $(a,c) \in f_2$. Como f_1 y f_2 son compatibles, $f_1 \cup f_2$ es una función. Además, como $(a,b),(a,c) \in f_1 \cup f_2$, $b = c$ y, por tanto, $\bigcup \mathcal{F}$ es una función.

Notemos que $x \in Dom(\bigcup \mathcal{F})$ si y sólo si existe y tal que $(x,y) \in \bigcup \mathcal{F}$, si y sólo si $(x,y) \in f$ para alguna $f \in \mathcal{F}$, si y sólo si $x \in \bigcup\{Dom(f) | f \in \mathcal{F}\}$. Por lo que $Dom(\bigcup \mathcal{F}) = \bigcup\{Dom(f) | f \in \mathcal{F}\}$. □

Lema 2.2. Sean $(A, \leqslant_A)$ y $(B, \leqslant_B)$ dos órdenes lineales, numerables, densos y sin extremos, h un isomorfismo parcial de A en B con dominio finito, $a \in A$ y $b \in B$. Entonces, existe un isomorfismo parcial $h_{a,b}$ tal que $h \subseteq h_{a,b}$, $a \in Dom(h_{a,b})$ y $b \in Ran(h_{a,b})$.

Demostración. Sea $h = \{(a_1,b_1),\ldots,(a_k,b_k)\}$ un isomorfismo parcial de A en B con dominio finito. Sin pérdida de generalidad, supongamos que $a_1 \leqslant_A a_2 \leqslant_A \ldots \leqslant_A a_k$ y $b_1 \leqslant_B b_2 \leqslant_B \ldots \leqslant_B b_k$. Consideremos los siguientes casos:

i) $a \in Dom(h), b \in Ran(h)$.

Basta tomar $h_{a,b} = h$, que es isomorfismo parcial por hipótesis.

ii) $a \notin Dom(h), b \in Ran(h)$.

Puesto que A es un orden lineal, denso y sin extremos, se tiene que $a \leqslant_A a_1$, $a_i \leqslant_A a \leqslant_A a_{i+1}$ para algún $1 \leqslant i < k$, o $a_k \leqslant a$. Tomemos $b_a \in B$ tal que:

- $b_a \leqslant_B b_1$ si $a \leqslant_A a_1$;
- $b_i \leqslant_B b_a \leqslant_B b_{i+1}$ si $a_i \leqslant_A a \leqslant_A a_{i+1}$;
- $b_k \leqslant_B b_a$ si $a_k \leqslant_A a$.

Tomando $h_{a,b} = h \cup \{(a,b_a)\}$, claramente se cumple que $h_{a,b}$ es un isomorfismo parcial tal que $a \in Dom(h_{a,b})$ y $b \in Ran(h_{a,b})$ por construcción.

iii) $a \in Dom(h), b \notin Ran(h)$.

Puesto que B es un orden lineal, denso y sin extremos, se tiene que $b \leqslant_B b_1$, $b_i \leqslant_B b \leqslant_B b_{i+1}$ para algún $1 \leqslant i < k$, o $b_k \leqslant_B b$. Tomemos $a_b \in A$ tal que:

- $a_b \leqslant_A a_1$ si $b \leqslant_B b_1$;
- $a_i \leqslant_A a_b \leqslant_A a_{i+1}$ si $b_i \leqslant_B b \leqslant_B b_{i+1}$;
- $a_k \leqslant_A a_b$ si $b_k \leqslant_B b$.

Tomando $h_{a,b} = h \cup \{(a_b, b)\}$, claramente se cumple que $h_{a,b}$ es un isomorfismo parcial tal que $a \in Dom(h_{a,b})$ y $b \in Ran(h_{a,b})$ por construcción.

iv) $a \notin Dom(h), b \notin Ran(h)$.

Tomando $h_{a,b} = h \cup \{(a, b_a), (a_b, b)\}$, donde a_b, b_a se construyen como en los dos casos anteriores, claramente se cumple que $h_{a,b}$ es un isomorfismo parcial tal que $a \in Dom(h_{a,b})$ y $b \in Ran(h_{a,b})$ por construcción.

□

Teorema 2.3. Cualesquiera dos órdenes lineales, numerables, densos y sin extremos son isomorfos.

Demostración. Sean $(A, \leqslant_A)$ y $(B, \leqslant_B)$ dos órdenes lineales, numerables, densos y sin extremos.

Sean $\{a_n | n \in \mathbb{N}\}$ y $\{b_n | n \in \mathbb{N}\}$ numeraciones sin repeticiones de A y B respectivamente.

Construyamos la siguiente sucesión de isomorfismos parciales compatibles por recursión:

$$h_0 = \{(a_0, b_0)\}$$
$$h_n = (h_{n-1})_{a_n, b_n}$$

Con $(h_{n-1})_{a_n,b_n}$, una extensión de h_{n-1} como la descrita en el lema 2.2.

Tomemos $h = \bigcup_{n \in \mathbb{N}} h_n$. Como h_n es un isomorfismo parcial de A en B para cada $n \in \mathbb{N}$, h es un isomorfismo de A en B.

□

2.1.2. Modelo estándar de la aritmética

A continuación definimos los modelos *estándar* y *no estándar* de la aritmética y mostramos que todos los modelos estándar son isomorfos entre sí.

Definición 2.11. I) Sea $\mathfrak{U}$ un modelo de la aritmética. Los elementos finitos o *estándar* de $\mathfrak{U}$ son aquellos de la forma $a = (n)^{\mathfrak{U}}$, para algún $n \in \mathbb{N}$; los elementos restantes se llaman *no estándar* o infinitos.

II) Para cualquier modelo $\mathfrak{U}$ de la aritmética, sea $Fin(\mathfrak{U}) = \{a \in D_{\mathfrak{U}} | a \text{ es finito}\} = \{(n)^{\mathfrak{U}} | n \in \mathbb{N}\}$.

III) Un modelo $\mathfrak{U}$ de la aritmética es *estándar* si carece de elementos *no estándar*. En otro caso es un modelo *no estándar*.

En particular, la estructura $\eta = \langle \mathbb{N}, s, +, \cdot, 0 \rangle$ es un modelo *estándar* de la aritmética. Aún más, el siguiente teorema muestra que cualquier modelo estándar de la aritmética es isomorfo a η.

Teorema 2.4. Para todo modelo estándar $\mathfrak{U}$ de la aritmética, existe un isomorfismo $f : \mathbb{N} \to D_{\mathfrak{U}}$.

Demostración. Sea $\mathfrak{U}$ un modelo estándar de la aritmética y definamos f como $f(n) = (s^n 0)^{\mathfrak{U}}$ para cada $n \in \mathbb{N}$.

Sean $n, m \in \mathbb{N}$ tales que $n \neq m$. Entonces, $\eta \vDash n \neq m$, por lo que $f(n) = (s^n 0)^{\mathfrak{U}} \neq (s^m 0)^{\mathfrak{U}} = f(m)$. Es decir, $f(n) \neq f(m)$, por lo que f es inyectiva.

Sea $a \in D_{\mathfrak{U}}$. Como $\mathfrak{U}$ es estándar, existe $n \in \mathbb{N}$ tal que $a = (n)^{\mathfrak{U}} = (s^n 0)^{\mathfrak{U}} = f(n)$. Es decir, existe $n \in \mathbb{N}$ tal que $a = f(n)$, por lo que f es sobreyectiva.

Por lo tanto, f es biyectiva.

Veamos que para cualesquiera $n, m \in \mathbb{N}$ se cumple que (1) $f(sn) = s^{\mathfrak{U}} f(n)$, (2) $f(n + m) = f(n) +^{\mathfrak{U}} f(m)$ y (3) $f(n \cdot m) = f(n) \cdot^{\mathfrak{U}} f(m)$.

(1) Sea $k = sn$. Entonces, $\eta \vDash k = sn$, por lo que $s^{\mathfrak{U}} f(n) = s^{\mathfrak{U}}(s^n 0)^{\mathfrak{U}} = (s^{n+1} 0)^{\mathfrak{U}} = (s^k 0)^{\mathfrak{U}} = f(k) = f(sn)$.

(2) Sea $k = n + m$. Entonces, $\eta \vDash k = n + m$, por lo que $f(n) +^{\mathfrak{U}} f(m) = (s^n 0)^{\mathfrak{U}} +^{\mathfrak{U}} (s^m 0)^{\mathfrak{U}} = (s^{n+m} 0)^{\mathfrak{U}} = (s^k 0)^{\mathfrak{U}} = f(k) = f(n + m)$.

(3) Sea $k = n \cdot m$. Entonces, $\eta \vDash k = n \cdot m$, por lo que $f(n) \cdot^{\mathfrak{U}} f(m) = (s^n 0)^{\mathfrak{U}} \cdot^{\mathfrak{U}} (s^m 0)^{\mathfrak{U}} = (s^{n \cdot m} 0)^{\mathfrak{U}} = (s^k 0)^{\mathfrak{U}} = f(k) = f(n \cdot m)$.

Por lo anterior, podemos concluir que f es un isomorfismo[1] y que η es isomorfo a cualquier modelo estándar de la aritmética.

□

2.1.3. Modelos no estándar de la aritmética

A continuación mostramos la existencia de modelos *no estándar* de la aritmética y analizamos su relación con el modelo estándar y su estructura.

Teorema 2.5. Existen modelos no estándar de la aritmética.

Demostración. Sean $\overline{L_{AR}} = L_{AR} \cup \{c\}$, con c un nuevo símbolo de constante y $AR' = AR \cup \{\neg(c = 0), \neg(c = s0), \ldots\}$. Veamos que AR' es satisfacible.

Consideremos $\Delta \subset AR'$ finito. Entonces existe $n \in \mathbb{N}$ tal que $\Delta \subseteq AR \cup \{\neg(c = 0), \neg(c = s0), \ldots, \neg(c = s^n 0)\}$. Sea η' la estructura construida a partir de η, en la que adicionalmente interpretamos a c como el número $n + 1$. Claramente, $\eta' \vDash \Delta$ y, por el teorema de compacidad, AR' es satisfacible.

Sean $\mathfrak{U}$ un modelo de AR' y $a = (c)^{\mathfrak{U}}$. Entonces, para todo $n \in \mathbb{N}$ se cumple que $\mathfrak{U} \vDash \neg(a = s^n 0)$. Por lo tanto, $\mathfrak{U}$ es un modelo no estándar de la aritmética. □

El teorema anterior muestra la existencia de los modelos no estándar de la aritmética. Los dos teoremas siguientes muestran cómo se relacionan estos modelos con el modelo estándar.

[1] Podemos llegar a esta conclusión, ya que toda función y relación recursiva se definen con base en las funciones iniciales (s, $+$ y $\cdot$).

Teorema 2.6. Ningún modelo no estándar de la aritmética es isomorfo a η.

Demostración. Sea $\mathfrak{U}$ un modelo no estándar y supongamos que existe un isomorfismo $f : \mathfrak{U} \to \mathbb{N}$. Como $\mathfrak{U}$ es un modelo no estándar, existe $a \in D_{\mathfrak{U}}$ tal que para todo $n \in \mathbb{N}$ se cumple que $a \neq (n)^{\mathfrak{U}}$.

Supongamos que $f(a) = k$. Entonces, debe suceder que $f(a) \neq f((k)^{\mathfrak{U}})$, pues en caso contrario se tendría que $a = f^{-1}(f(a)) = f^{-1}(f((k)^{\mathfrak{U}})) = (k)^{\mathfrak{U}}$, por lo que $\eta \vDash f(a) \neq k$, o $\eta \vDash k \neq k$!

Por lo tanto, no hay un isomorfismo entre $\mathfrak{U}$ y η. □

Teorema 2.7. $\eta = \langle \mathbb{N}, +, \cdot, s, 0 \rangle$ es un *modelo primo*[2] de la aritmética.

Demostración. Debemos demostrar que η admite un encaje elemental en cualquier modelo $\mathfrak{U}$ de la aritmética. Para ello consideremos los siguientes dos casos:
Caso 1. $\mathfrak{U}$ es un modelo estándar.

Se cumple por el teorema 2.4 y por la definición 2.11.

Caso 2. $\mathfrak{U}$ es un modelo no estándar.

Consideremos el submodelo $\mathfrak{U}_f = \langle Fin(\mathfrak{U}), +_f, \cdot_f, s_f, (0)^{\mathfrak{U}} \rangle^{\ddagger}$. Notemos que $\mathfrak{U}_f \leqslant_e \mathfrak{U}$ por definición y veamos que η es isomorfo a $\mathfrak{U}_f$.

Veamos que $i : \mathbb{N} \to Fin(\mathfrak{U})$ definida como $i(n) = (s^n 0)^{\mathfrak{U}}$ es un isomorfismo.

Sean $n, m \in \mathbb{N}$ tales que $n \neq m$. Entonces, $\eta \vDash n \neq m$, por lo que $i(n) = (s^n 0)^{\mathfrak{U}} \neq (s^m 0)^{\mathfrak{U}} = i(m)$. Es decir, $i(n) \neq i(m)$, por lo que i es inyectiva.

Notemos que i es sobreyectiva por definición de $Fin(\mathfrak{U})$, por lo que basta ver que para cada $n, m \in \mathbb{N}$ sucede que (I) $i(sn) = s_f i(n)$, (II) $i(n + m) = i(n) +_f i(m)$ y (III) $i(n \cdot m) = i(n) \cdot_f i(m)$.

[2] Véase la definición 2.6.

[‡] $+_f$, $\cdot_f$ y s_f representan las restricciones de $+^{\mathfrak{U}}$, $\cdot^{\mathfrak{U}}$ y $s^{\mathfrak{U}}$ a $Fin(\mathfrak{U})$.

(I) Sea $k = sn$. Entonces, $\eta \vDash k = sn$, por lo que $s_f i(n) = s^{\mathfrak{U}} i(n) = s^{\mathfrak{U}}(s^n 0)^{\mathfrak{U}} = (s^{n+1}0)^{\mathfrak{U}} = (s^k 0)^{\mathfrak{U}} = i(k) = i(sn)$.

(II) Sea $k = n + m$. Entonces, $\eta \vDash k = n + m$, por lo que $i(n) +_f i(m) = i(n) +^{\mathfrak{U}} i(m) = (s^n 0)^{\mathfrak{U}} +^{\mathfrak{U}} (s^m 0)^{\mathfrak{U}} = (s^{n+m}0)^{\mathfrak{U}} = (s^k 0)^{\mathfrak{U}} = i(k) = i(n+m)$.

(III) Sea $k = n \cdot m$. Entonces, $\eta \vDash k = n \cdot m$, por lo que $i(n) \cdot_f i(m) = i(n) \cdot^{\mathfrak{U}} i(m) = (s^n 0)^{\mathfrak{U}} \cdot^{\mathfrak{U}} (s^m 0)^{\mathfrak{U}} = (s^{n \cdot m}0)^{\mathfrak{U}} = (s^k 0)^{\mathfrak{U}} = i(k) = i(n \cdot m)$.

Por lo tanto, η es isomorfo a $\mathfrak{U}_f$.

Es así que η admite un encaje elemental en cualquier modelo $\mathfrak{U}$ de la aritmética. Es decir, η es un modelo primo de esta última.

□

A continuación mostramos varias propiedades de los modelos no estándar de la aritmética, lo que nos permitirá tener una mejor idea de cómo están estructurados.

Teorema 2.8. Sea $\mathfrak{U}$ un modelo no estándar de la aritmética. Si $a \in Fin(\mathfrak{U})$ y $b \in D_{\mathfrak{U}} \backslash Fin(\mathfrak{U})$, $a <^{\mathfrak{U}} b$.

Demostración. Sea x una variable y definamos para cada $n \in \mathbb{N}$, ψ_n una fórmula con variable libre x como sigue:

- ψ_0 es la fórmula $x \neq 0$ y
- $\psi_{n+1} = (\psi_n \wedge x \neq s^{n+1}0)$.

Es decir, $\psi_n = (x \neq 0 \wedge x \neq s0 \wedge \cdots \wedge x \neq s^n 0)$.

Notemos que para cualquier $n \in \mathbb{N}$, sucede que $\eta \vDash \forall x(\psi_n \Leftrightarrow x > s^n 0)$.

Sea $a = (s^n 0)^{\mathfrak{U}}$, para algún $n \in \mathbb{N}$.

Como $\eta \vDash \forall x(\psi_n \Leftrightarrow x > s^n 0)$, en particular sucede $\mathfrak{U} \vDash \forall x(\psi_n \Leftrightarrow x > s^n 0)$.

Como $\mathfrak{U} \vDash (b \neq 0 \wedge b \neq s0 \wedge \cdots \wedge b \neq s^n 0)$, podemos deducir que $\mathfrak{U} \vDash (b > s^n 0)$. Es decir, $\mathfrak{U} \vDash b > a$. □

Teorema 2.9. Sean $\phi(x)$ una fórmula de L_{AR} y $\mathfrak{U}$ un modelo no estándar de la aritmética. Entonces:

a) $\mathfrak{U} \vDash \phi[a]$ para toda $a \in Fin(\mathfrak{U})$ si y sólo si $\mathfrak{U} \vDash \phi[b]$ para toda $b \in D_{\mathfrak{U}}$.

b) $\mathfrak{U} \vDash \phi[b]$ para toda $b \in D_{\mathfrak{U}} \backslash Fin(\mathfrak{U})$ si y sólo si $\mathfrak{U} \vDash \phi[a]$ para toda $a \in Fin(\mathfrak{U})$, salvo una cantidad finita.

c) $\mathfrak{U} \vDash \phi[b]$ para alguna $b \in D_{\mathfrak{U}} \backslash Fin(\mathfrak{U})$ si y sólo si $\mathfrak{U} \vDash \phi[a]$ para una infinidad de $a \in Fin(\mathfrak{U})$.

Demostración. a) Notemos que se cumple la cadena de equivalencias :

$\mathfrak{U} \vDash \phi[a]$ para toda $a \in Fin(\mathfrak{U}) \Leftrightarrow$ para toda $n \in \mathbb{N}, \mathfrak{U} \vDash \phi(s^n 0) \Leftrightarrow$ para toda $n \in \mathbb{N}, \eta \vDash \phi(s^n 0) \Leftrightarrow$ para toda $n \in \mathbb{N}, \eta \vDash \phi(n) \Leftrightarrow \eta \vDash \forall x \phi(x) \Leftrightarrow \mathfrak{U} \vDash \forall x \phi(x) \Leftrightarrow \mathfrak{U} \vDash \phi[b]$ para toda $b \in D_{\mathfrak{U}}$.

b) Notemos que se cumple la cadena de implicaciones:

Para toda $b \in D_{\mathfrak{U}} \backslash Fin(\mathfrak{U}), \mathfrak{U} \vDash \phi(b) \Rightarrow$ para toda $b \in D_{\mathfrak{U}} \backslash Fin(\mathfrak{U}), \mathfrak{U} \vDash (\forall c(x > b \Rightarrow \phi(x))) \Rightarrow$ existe $b \in D_{\mathfrak{U}}$ tal que $\mathfrak{U} \vDash (\forall x(x > b \Rightarrow \phi(x))) \Rightarrow \mathfrak{U} \vDash \exists y \forall x(x > y \Rightarrow \phi(x)) \Rightarrow \eta \vDash \exists y \forall x(x > y \Rightarrow \phi(x)) \Rightarrow$ existe $k \in \mathbb{N}$ tal que para toda $n > k, \eta \vDash \phi(n)$.

Recíprocamente se cumple la cadena de implicaciones:

Para toda $n > k, \eta \vDash \phi(n) \Rightarrow \eta \vDash \forall x(x > s^k 0 \Rightarrow \phi(x)) \Rightarrow \mathfrak{U} \vDash \forall x(x > s^k 0 \Rightarrow \phi(x)) \Rightarrow$ para toda $b \in D_{\mathfrak{U}} \backslash Fin(\mathfrak{U}), \mathfrak{U} \vDash \phi[b]$.

c) Se cumple por b).

□

Corolario 2.1. La expresión "x es finito" es indescriptible mediante una fórmula de L_{AR}.

Demostración. Si existiera $\phi(x)$ una fórmula tal que para toda $a \in D_{\mathfrak{U}}$, $\mathfrak{U} \vDash \phi[a]$ si y sólo si $a \in Fin(\mathfrak{U})$, se tendría una contradicción con el teorema 2.9 a). Por lo tanto, dicha fórmula no puede existir. □

Definición 2.12. Sea $S = s^{\mathfrak{U}} : \mathfrak{U} \to \mathfrak{U}$. Definamos la función predecesor $p : \mathfrak{U} \to \mathfrak{U}$ como

$$p(a) = \begin{cases} b, & \text{si y sólo si } \mathfrak{U} \vDash a > 0, \text{ y } S(b) = a \\ 0, & \text{si } a = (0)^{\mathfrak{U}}. \end{cases}$$

En forma similar, podemos extender a $\mathfrak{U}$ las demás funciones y relaciones definibles de η.

Definición 2.13. Sean $x, y \in D_{\mathfrak{U}}$. Definimos la relación $\sim$ como

$$x \sim y \Leftrightarrow \exists n \in \mathbb{N}(S^n x = y \vee S^n y = x).$$

Proposición 2.1. 1) $\sim$ es una relación de equivalencia.

2) $0 \sim x$ si y sólo si $x \in Fin(\mathfrak{U})$ (por lo que $\sim$ no es definible en L_{AR}).

3) Si $a < b < c$ y $a \sim c$, entonces $a \sim b$.

Demostración. 1) Veamos que $\sim$ es reflexiva, simétrica y transitiva.

Reflexividad.

Sea $x \in D_{\mathfrak{U}}$ y notemos que $S^0 x = x$, por lo que $\exists n \in \mathbb{N}(S^n x = x \vee S^n x = x)$. Por lo tanto, $x \sim x$.

Simetría.

Sean $x, y \in D_{\mathfrak{U}}$ tales que $x \sim y$. Entonces, $\exists n \in \mathbb{N}(S^n x = y \vee S^n y = x)$, que es equivalente a $\exists n \in \mathbb{N}(S^n y = x \vee S^n x = y)$, por lo que $y \sim x$.

Transitividad.

Sean $x, y, z \in D_{\mathfrak{U}}$ tales que $x \sim y$ e $y \sim z$. Entonces, $\exists n \in \mathbb{N}(S^n x = y \vee S^n y = x)$ y $\exists m \in \mathbb{N}(S^m y = z \vee S^m z = y)$. Consideremos los siguientes casos:

Caso 1. $S^n x = y \wedge S^m y = z$.

Entonces, $S^{n+m} x = z$, por lo que $\exists k \in \mathbb{N}(S^k x = z \vee S^k z = x)$. Por lo tanto, $x \sim z$.

Caso 2. $S^n x = y \wedge S^m z = y$.

Si $n \leqslant m$, $S^{m-n}x = z$; en caso contrario, $S^{n-m}z = x$, por lo que $\exists k \in \mathbb{N}(S^k x = z \vee S^k z = x)$. Por lo tanto, $x \sim z$.

Caso 3. $S^n y = x \wedge S^m y = z$.

Análogo al caso 2.

Caso 4. $S^n y = x \wedge S^m z = y$.

Análogo al caso 1.

Por lo tanto, $\sim$ es una relación de equivalencia.

2) Sea $x \in D_{\mathfrak{U}}$ tal que $0 \sim x$. Entonces, $\exists n \in \mathbb{N}(S^n 0 = x \vee S^n x = 0)$ y como $\eta \vDash \neg\exists y(Sy = 0)$, sólo puede suceder que $S^n 0 = x$, por lo que $x \in Fin(\mathfrak{U})$.

Notemos, además, que si existiera una fórmula $\phi(x, y)$ de L_{AR} tal que $\mathfrak{U} \vDash \phi[a, b] \Leftrightarrow (a \sim b)$, se cumpliría que $a \in Fin(\mathfrak{U}) \Leftrightarrow \mathfrak{U} \vDash a \sim 0$, lo que contradice al corolario 2.1.

3) Notemos que $\mathfrak{U} \vDash S^n a = c$ para algún $n \in \mathbb{N}$, por definición del orden y de $\sim$. Puesto que $\eta \vDash \forall x, y, z[x < y < z \wedge s^n x = z \Rightarrow (y = sx \vee y = s^2 a \vee \cdots \vee y = s^{n-1}x)]$, se cumple que $\mathfrak{U} \vDash a < b < c \wedge S^n a = c \Rightarrow (b = Sa \vee b = S^2 a \vee \cdots \vee b = S^{n-1}a)$.

Luego, $\exists k \in \mathbb{N}(S^k a = b)$ y, por lo tanto, $a \sim b$.

□

Definición 2.14. 1) Dado $a \in D_{\mathfrak{U}}$, sea $a/\sim = \{b \in D_{\mathfrak{U}} | a \sim b\}$.

2) Sea $\mathfrak{U}/\sim = \{a/\sim | a \in D_{\mathfrak{U}}\}$.

Definición 2.15. Sean $a/\sim, b/\sim \in \mathfrak{U}/\sim$. Podemos definir un orden en $\mathfrak{U}/\sim$ como $a/\sim < b/\sim$ si y sólo si $a \nsim b$, y $a < b$.

Teorema 2.10. Sea $a \in D_{\mathfrak{U}}$.Entonces:

I) Si $a \in Fin(\mathfrak{U})$, entonces $a/\sim = Fin(\mathfrak{U})$.

II) Si $a \in D_{\mathfrak{U}} \backslash Fin(\mathfrak{U})$, entonces $\langle a/\sim, <, S_\sim \rangle$ es isomorfo a $\langle \mathbb{Z}, <, \sigma \rangle$, donde $S_\sim$ es la restricción de S a $a/\sim$ y σ es la función sucesor de los números enteros.

Demostración. Sea $a \in D_{\mathfrak{U}}$.

I) Por la proposición 2.1.2), $a \in Fin(\mathfrak{U})$ si y sólo si $a \sim 0$, por lo que $0 \in a/\sim$. Como $\sim$ es transitiva, se tiene que $\forall x(x \sim 0 \Rightarrow x \in a/\sim)$, por lo que $Fin(\mathfrak{U}) \subseteq a/\sim$. Como $a/\sim \subseteq Fin(\mathfrak{U})$ por definición de $a/\sim$, se tiene que $a/\sim = Fin(\mathfrak{U})$.

II) Notemos que $a/\sim = \{b \in D_{\mathfrak{U}} | \exists n \in \mathbb{N}[b = p^n(a) \vee b = S^n a]\}$, por definición de $\sim$.

Definamos $f : \mathbb{Z} \to a/\sim$ como

$$f(k) = \begin{cases} S^k a & \text{si } k \geqslant 0. \\ p^{-k}(a) & \text{si } k < 0. \end{cases}$$

Veamos que f respeta el orden y la función sucesor.

- Sean $x, y \in \mathbb{Z}$ tales que $x < y$ y consideremos los siguientes casos:

  Caso 1. $x < y < 0$.

  Entonces, $f(x) = p^{-x}(a)$ y $f(y) = p^{-y}(a)$. Como $-y < -x$, $p^{-x}(a) < p^{-y}(a)$ por definición del predecesor. Es decir, $f(x) < f(y)$.

  Caso 2. $x < 0 \leqslant y$.

  Entonces, $f(x) = p^{-x}(a)$ y $f(y) = S^y a$. Como $p^{-x}(a) < a \leqslant S^y a$ por definición de sucesor y predecesor, $f(x) < f(y)$.

  Caso 3. $0 < x < y$.

  Entonces, $f(x) = S^x a$ y $f(y) = S^y a$. Como $x < y$, $S^x a < S^y a$ por definición del sucesor. Es decir, $f(x) < f(y)$.

- Sea $x \in \mathbb{Z}$ y consideremos los siguientes casos:

  Caso 1. $\sigma x < 0$.
  Entonces, $f(\sigma x) = p^{-\sigma x}(a) = p^{-x-1}(a) = Sp^{-x}(a) = Sf(x)$ por definición de predecesor. Es decir, $f(\sigma x) = Sf(x)$.

  Caso 2. $\sigma x \geqslant 0$.
  Entonces, $f(\sigma x) = S^{\sigma x}a = S^{x+1}a = SS^x a = Sf(x)$ por definición de sucesor. Es decir, $f(\sigma x) = Sf(x)$.

Por lo tanto, $\langle a/\sim, <, S_\sim\rangle$ es isomorfo a $\langle \mathbb{Z}, <, \sigma\rangle$.

□

Teorema 2.11. Sea $\mathfrak{U}$ un modelo no estándar de la aritmética. Entonces:

1) Para cualesquiera $a, b \in D_{\mathfrak{U}}$, si $a/\sim < b/\sim$, entonces existe un elemento $c \in D_{\mathfrak{U}}$ tal que $a/\sim < c/\sim < b/\sim$.

2) $\mathfrak{U}' = \langle \mathfrak{U}/\sim, <\rangle$ tiene un primer elemento, pero no segundo elemento ni último elemento.

Demostración. 1) Sea $a < b$. Como $\mathfrak{U} \vDash \forall x, y \exists z(z+z = x+y \vee z+z = x+Sy)$, existe $c \in D_{\mathfrak{U}}$ tal que $\mathfrak{U} \vDash c + c = a + b$ ó $\mathfrak{U} \vDash c + c = a + Sb$.

Consideremos el caso en que $c + c = a + b$ (en el otro caso se procede de forma análoga, por lo que lo omitiremos). Por lo tanto, c es la media aritmética de a y b. Veamos que $a/\sim < c/\sim$ y que $c/\sim < b/\sim$.

a) $a/\sim < c/\sim$.

Como $\mathfrak{U} \vDash \forall x, y, z(x \leqslant y \wedge z + z = x + y \Rightarrow x \leqslant z)$, tenemos que $a \leqslant c$ y por tanto, $a/\sim \leqslant c/\sim$.

Supongamos que $a/\sim = c/\sim$. Entonces, existe $m \in \mathbb{N}$ tal que $\mathfrak{U} \vDash S^m a = c$. Pero para toda $n \in \mathbb{N}$, $\mathfrak{U} \vDash \forall x, y, z(x+y = z+z \wedge z = S^n x \Rightarrow y = S^{2n}x)$, por lo que $\mathfrak{U} \vDash b = S^{2n}a$ y, por tanto, $a \sim b$!

Por lo tanto, $a/\sim < c/\sim$.

b) $c/\sim < b/\sim$.

Como $\mathfrak{U} \vDash \forall x, y, x(x \leqslant y \wedge z + z = x + y \Rightarrow z \leqslant y)$, tenemos que $c \leqslant b$ y por tanto, $c/\sim \leqslant b/\sim$.

Supongamos que $c/\sim = b/\sim$. En tal caso, existe $m \in \mathbb{N}$ tal que $\mathfrak{U} \vDash S^m c = b$. Pero para toda $n \in \mathbb{N}$, $\mathfrak{U} \vDash \forall x, y, z(x+y = z+z \wedge y = S^n z \Rightarrow z = S^n x)$, por lo que $\mathfrak{U} \vDash c = S^n a$ y, por tanto, $a \sim c$!

Por lo tanto, $c/\sim < b/\sim$.

2) Como $\mathfrak{U} \vDash \forall x \neg(x < 0)$, $Fin(\mathfrak{U}) = 0/\sim$ es el primer elemento de $\mathfrak{U}'$.

Por 1), para cualquier $b \in D_{\mathfrak{U}} \backslash Fin(\mathfrak{U})$, existe $c \in D_{\mathfrak{U}}$ tal que $Fin(\mathfrak{U}) < c/\sim < b/\sim$, por lo que no puede haber un segundo elemento.

Como $\mathfrak{U} \vDash \forall x(x + x = S^n x \Rightarrow x = S^n 0)$, $a + a$ no puede estar en la misma clase de equivalencia que a, para todo $a \in D_{\mathfrak{U}} \backslash Fin(\mathfrak{U})$. Así que $\mathfrak{U} \vDash a + a > a$ y, por tanto, $\mathfrak{U}'$ no puede tener último elemento.

□

Con base en los teoremas 2.3, 2.11 y 2.10, podemos concluir que para cualquier modelo no estándar de la aritmética $\mathfrak{U}$, $\mathfrak{U}' = \langle \mathfrak{U}/\sim, < \rangle$ tiene tipo de orden $\omega + (\omega + \omega^*) \cdot \eta$, donde ω es el tipo de orden de $\mathbb{N}$, $\omega + \omega^*$ es el tipo de orden de $\mathbb{Z}$ y η es el tipo de orden de $\mathbb{Q}$.

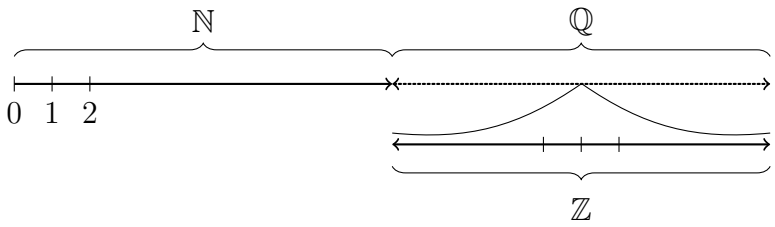


Figura 2.1: Representación gráfica de $\omega + (\omega + \omega^*) \cdot \eta$. Intuitivamente, es como si "pegáramos" la recta racional al final de la recta natural, sustituyendo cada número racional por una recta entera.

2.2. Definiciones, diferencias y similitudes

Definición 2.16. Sea T una teoría axiomática de primer orden. Decimos que T es *consistente* si para cualquier fórmula B de L_T no sucede que $\vdash_T B$ y $\vdash_T \neg B$.[3]

Definición 2.17. Sea T una teoría axiomática de primer orden con numerales y operadores para la negación y cuantificación universal. Decimos que T es ω-*consistente* si para toda fórmula $A(x)$ de L_T, sucede que si $\vdash_T A(\bar{k})$ para toda $k \in \mathbb{N}$, entonces $\nvdash_T \exists x \neg A(x)$.

Definición 2.18. Sea T una teoría axiomática de primer orden que contiene a AR. Decimos que T es *correcta* si sus teoremas son verdaderos en el *modelo estándar* de la aritmética.

Intuitivamente, una teoría es consistente cuando no es contradictoria; es ω-consistente cuando no puede probar enunciados existenciales que sólo son válidos en los modelos no estándar, y es correcta cuando no prueba enunciados aritméticos falsos en el modelo estándar.

Si bien hay cierta similitud entre las definiciones anteriores, también hay cierta diferencia entre ellas, como muestran los siguientes teoremas y ejemplos.

Teorema 2.12. Si una teoría aritmética de primer orden es correcta, entonces es consistente.

Demostración. Sea T una teoría aritmética de primer orden correcta, η el modelo estándar de la aritmética y supongamos que T no es consistente. Entonces, existe una fórmula B de L_T tal que $\vdash_T B$ y $\vdash_T \neg B$, por lo que al ser T correcta se tiene que $\eta \vDash B$ y $\eta \vDash \neg B$. Por lo tanto, B es verdadera y a la vez falsa en η!

Luego, dicha B no puede existir y, por lo tanto, T es consistente. □

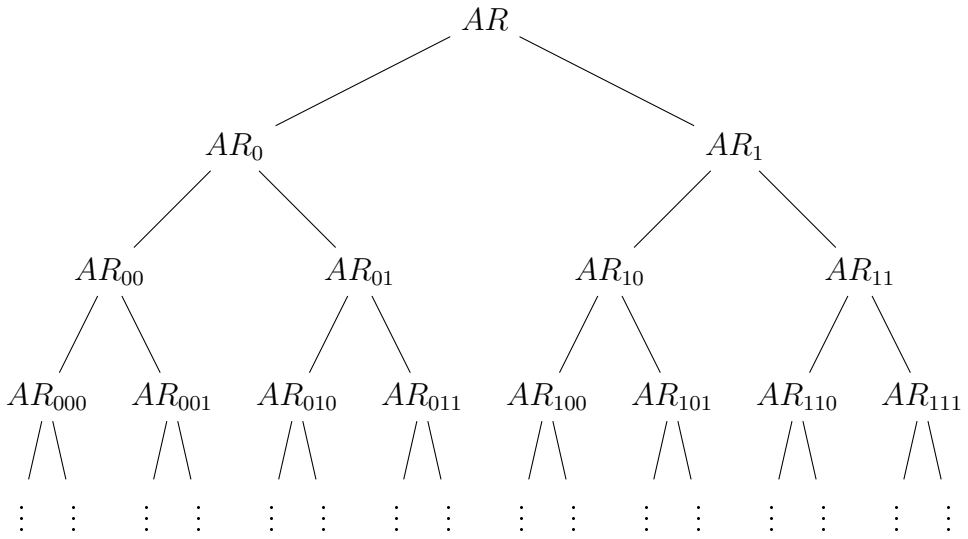
Ejemplo 2.1. Supongamos que AR, la aritmética recursiva, es consistente y consideremos a G su enunciado de Gödel. Por el primer teorema de Gödel, sabemos que G es verdadero pero no demostrable en AR, por lo que $AR_1 = AR \cup \{\neg G\}$ es consistente. Sin embargo, AR_1 es incorrecta, pues el

[3] Alternativamente se puede definir que una teoría es consistente si existe una fórmula B de L_T tal que $\nvdash_T B$.

enunciado $\neg G$ es falso en el modelo estándar de AR.

Por otro lado, si en vez de considerar $AR_1 = AR \cup \{\neg G\}$, consideramos $AR_0 = AR \cup \{G\}$, tenemos que AR_0 sí es correcta, ya que el enunciado G es verdadero en el modelo estándar de AR. Puesto que el teorema de Gödel se cumple en cualquier extensión de AR, existirán dos enunciados G_0 y G_1 tales que $\nvdash_{AR_0} G_0$, $\nvdash_{AR_0} \neg G_0$, $\nvdash_{AR_1} G_1$ y $\nvdash_{AR_1} \neg G_1$.

Procediendo de la misma manera como lo hicimos con AR, podemos construir las extensiones $AR_{00} = AR_0 \cup \{G_0\}$, $AR_{01} = AR_0 \cup \{\neg G_0\}$, $AR_{10} = AR_1 \cup \{G_1\}$ y $AR_{11} = AR_1 \cup \{\neg G_1\}$. De las cuales, sólo AR_{00} es correcta. Si repetimos el procedimiento, podemos generar las siguientes extensiones:



De las cuales, sólo las que se encuentran en la diagonal izquierda (*i.e.* aquellas cuyo subíndice está formado únicamente por ceros) son correctas.

Por lo tanto, el hecho de que una teoría sea consistente no necesariamente implica que sea correcta.

Teorema 2.13. Si una teoría axiomática de primer orden es ω-consistente, entonces es consistente.

Demostración. Sean T una teoría axiomática de primer orden ω-consistente, $A(x)$ una fórmula de L_T que tiene a x como única variable libre y $B(x) \equiv_{def} A(x) \wedge \neg A(x)$. Entonces, $\neg B(\bar{n})$ es instancia de una tautología, por lo que $\vdash_T \neg B(\bar{n})$ para todo número natural n. Luego, por ω-consistencia, $\nvdash_T \exists x B(x)$. Por lo tanto, T es consistente. □

Ejemplo 2.2. Consideremos a AR, la aritmética recursiva, y a $CON \equiv_{def} \forall x \neg \overline{PR}(x, \ulcorner 0 = 1 \urcorner)$. El enunciado CON interpretado en la metateoría a través del código de Gödel, afirma la consistencia de AR. Por el segundo teorema de Gödel, sabemos que si AR es consistente entonces $T = AR \cup \{\neg CON\}$ también lo es. Sin embargo, T no es ω-consistente, pues para todo número natural n, $\vdash_T \neg \overline{PR}(\bar{n}, \ulcorner 0 = 1 \urcorner)$ y $\vdash_T \exists x \overline{PR}(x, \ulcorner 0 = 1 \urcorner)$.

Por lo tanto, el que una teoría sea consistente no necesariamente implica que sea ω-consistente.

El teorema 2.12 junto con el ejemplo 2.1 muestran que el concepto de correctud es más "fuerte" que el de consistencia. De la misma manera, el teorema 2.13 junto con el ejemplo 2.2 muestran que el concepto de ω-consistencia es más "fuerte" que el de consistencia.

El objetivo de este capítulo fue definir los conceptos de consistencia, ω-consistencia y completud, además de hacer notar sus similitudes y diferencias. Para lograrlo, fue necesario introducir algunos conceptos y resultados previos propios de la teoría de modelos y de la teoría de ordinales. En el siguiente capítulo definiremos el concepto de complejidad Kolmogórov y resaltaremos algunas de sus características.

Capítulo 3

Complejidad Kolmogórov

Dentro de la teoría matemática de la información, se conoce como *entropía* a cualquier cantidad que permita medir la indeterminación de una variable. Usualmente se define de la siguiente manera:

Definición 3.1. Supongamos que una variable x es capaz de tomar valores en un conjunto finito X que contiene N elementos. Definimos la *entropía* de x como $H(x) = \log_2 N$. Cuando tomamos un valor específico, digamos $x = a$, entonces eliminamos dicha entropía y obtenemos la "información" contenida en a, dada por $I(a) = \log_2 N$.

En 1968 A. N. Kolmogórov presentó un artículo titulado *Tres enfoques para la definición cuantitativa de información* [9], en el cual define una nueva forma de medir la "información" contenida en un número a través de la cantidad de información requerida para generarlo mediante un algoritmo. En este capítulo presentamos dicha definición, primero de manera intuitiva y después de manera formal, mediante tres definiciones distintas (pero equivalentes): a través de las funciones recursivas, los lenguajes de programación y las máquinas de Turing.

3.1. Idea intuitiva

Consideremos los siguientes algoritmos que aproximan π y la raíz cuadrada de un número natural x.

Algoritmo de Gauss-Legendre para aproximar π.	Algoritmo babilónico para aproximar $\sqrt{x}$.
1. Tómense $a_0 = 1$, $b_0 = \frac{1}{\sqrt{2}}$, $p_0 = 1$ y $t_0 = \frac{1}{4}$.	1. Tómense dos números b y h tales que $bh = x$.
2. Defínase $a_{n+1} = \frac{a_n + b_n}{2}$.	2. Si $h \approx b$, vaya al paso 6; si no, vaya al paso 3.
3. Defínase $b_{n+1} = \sqrt{a_n b_n}$.	3. Hágase $b = \frac{h + b}{2}$.
4. Defínase $p_{n+1} = 2p_n$.	4. Hágase $h = \frac{x}{b}$.
5. Defínase $t_{n+1} = t_n - p_n(a_n - a_{n+1})^2$.	5. Vaya al paso 2.
6. $\pi \approx \frac{(a_n + b_n)^2}{4t_n}$.	6. $\sqrt{x} \approx b$.

Se antoja que una forma de medir la cantidad de "información" contenida por estos números podría ser a través de la cantidad de renglones que poseen los algoritmos que los generan. Con base en esta idea, tendríamos que $K(\pi) = 6$ y $K(\sqrt{x}) = 6$. Sin embargo, esta idea resulta poco útil, ya que la "información" contenida por $\sqrt{x}$ debería variar dependiendo de si el resultado es un número entero, racional o irracional. Además, intuitivamente, cualquier número trascendente debería poseer una mayor "información" que cualquier número algebraico.

3.2. Definición formal

A continuación daremos tres definiciones de la complejidad Kolmogórov de un número x.

Definición 3.2. Sean $\{\varphi_n\}_{n\in\mathbb{N}}$ una enumeración de las funciones recursivas[1] de un argumento y x un número. Definimos la complejidad Kolmogórov de x como $K(x) = \mu n(\varphi_n(0) = x)$.

Definición 3.3. Sean $\{\mathcal{M}_i\}_{i\in\mathbb{N}}$ una enumeración de las máquinas de Turing[2] de un argumento, $\mathcal{U}(a, b)$ una máquina de Turing universal y x un número. Definimos la complejidad Kolmogórov de x como $K(x) = \mu i(\mathcal{U}(i, \varepsilon)\downarrow x)$. Es decir, es el menor i tal que $\mathcal{M}_i(\varepsilon)\downarrow x$.

Definición 3.4. Sea $\mathfrak{L}$ un lenguaje de programación fijo. Definimos la complejidad Kolmogórov (con respecto a $\mathfrak{L}$) de x ($K(x)$) como la longitud (en bits) del programa más pequeño que imprime x y se detiene.[3]

La idea detrás de estas definiciones es poder precisar la noción de "información" contenida por un número de forma simple. Asimismo, Kolmogórov define una nueva clase de números a los que llama *aleatorios*, que son aquellos números x tales que $K(x) > x$. Es decir, un número es aleatorio cuando se requiere una gran cantidad de información para generarlo.

Notemos que con estas definiciones sí se cumple que $K(\pi) > K(\sqrt{x})$. Para esto, consideremos la definición 3.4 y los siguientes programas que los generan:

Programa que genera π.

```
#include<stdio.h>
#include<math.h>
int main(void){
  int n;
  long double a[6],b[6],p[6],t[6],pi=0.0;
  a[0]=1;
  b[0]=1/sqrt(2.0);
  p[0]=1;
  t[0]=1.0/4.0;
  for(n=1;n<6;n++){
    a[n]=(a[n-1]+b[n-1])/2.0;
    b[n]=sqrt(a[n-1]*b[n-1]);
    p[n]=2.0*p[n-1];
```

[1]Para un estudio detallado de las funciones recursivas véase [18],[14].

[2]Para un estudio detallado de las máquinas de Turing véase [1].

[3]Si bien estas tres definiciones son equivalentes, en los siguientes capítulos utilizaremos únicamente la definición 3.3, pues es la que mejor se adecua al propósito de este trabajo.

```
    t[n]=t[n-1]*pow(a[n-1]-a[n],2));
    pi=pow(a[n]+b[n],2)/(4.0*t[n]);}
  printf("Pi=%.6Lf",pi);
  return 0;}
```

Programa que genera $\sqrt{x}$.

```
#include<stdio.h>
int main(void){
  int n;
  float x,y=1,e=0.0000001;
  printf("n=");
  scanf("%d",&n);
  x=n;
  while(x-y>e){
    x=(x+y)/2;
    y=x/n;}
  printf("Raiz de %d=%.6f",n,x);
  return 0;}
```

De acuerdo con los programas anteriores y la definición 3.4, se tiene que $K(\pi) \leqslant 1456$ y $K(\sqrt{x}) \leqslant 840$, por lo que $K(\pi) > K(\sqrt{x})$. Además, también se cumple que $K(\pi) > \pi$, por lo que π resulta ser un número aleatorio. Por otro lado, la aleatoriedad de $\sqrt{x}$ dependerá del valor de x.

Con el propósito de clarificar las definiciones anteriores y mostrar que la complejidad de un número puede variar considerablemente, según la definición de complejidad Kolmogórov utilizada, presentamos el siguiente ejemplo:

Ejemplo 3.1. Consideremos el número 2 y aproximemos su complejidad Kolmogórov mediante cada una de las tres definiciones anteriores.

- Consideremos la numeración de Gödel de L_{AR}

$\forall$	$\neg$	$\rightarrow$	$($	$)$	R	C	$=$	0	s	$K_{0,0}$	x_n	$P_{n,k}(1 \leqslant k \leqslant n)$
$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
1	3	5	7	9	11	13	15	17	19	21	$23+4n$	$17+4(t_n+k)^{\ddagger}$

y la función recursiva ssx_0. Entonces, de acuerdo con la numeración de Gödel anterior, tenemos que $\gamma ssx_0 = 2^{19} \cdot 3^{19} \cdot 5^{23} = 7.26413416875 \times 10^{30}$. Luego, como $ss0 = 2$, tenemos que $K(2) \leqslant 7.26413416875 \times 10^{30}$.

$^{\ddagger}t_n$ es el enésimo número triangular: $t_n = 1 + 2 + \ldots + n$.

- Consideremos la máquina de Turing $\mathcal{M}$ descrita en el ejemplo 1.3, cuyo número de descripción es $31332253117311332263111^{\dagger} \approx 3.1 \times 10^{22}$. Como $\mathcal{M}\downarrow 10$ y 2 en sistema binario es 10, tenemos que $K(2) \lesssim 3.1 \times 10^{22}$.

- Consideremos el lenguaje de programación C y el programa

```
#include<stdio.h>
int main(void){
    printf(“2”);
    return 0;}
```

cuya ejecución simplemente imprime el número 2 y que tiene una longitud de 464 bits (58 bytes). Entonces, de acuerdo con nuestra definición, tenemos que $K(2) \leqslant 464$. Como puede verse en este ejemplo, la complejidad Kolmogórov de un número puede diferir, por varios órdenes de magnitud, según la definición utilizada.

El concepto de complejidad Kolmogórov, previamente definido, tiene diversos resultados y aplicaciones interesantes. Algunos de ellos son el teorema de Chaitin, que enunciaremos y demostraremos en el siguiente capítulo, el hecho de que el conjunto de números *no aleatorios* es un conjunto simple(*i.e.* es recursivamente numerable y su complemento es un conjunto infinito sin subconjuntos recursivamente numerables)[6], o su aplicación a la inteligencia artificial en tests alternativos al de Turing[7].

En este capítulo definimos el concepto de complejidad Kolmogórov tanto de manera intuitiva como de manera formal. Hicimos notar cómo la complejidad de un número puede diferir considerablemente, según la definición utilizada, y mencionamos algunas de sus aplicaciones. En el siguiente capítulo explicaremos brevemente el primer teorema de Gödel y demostraremos el teorema de Chaitin, que tendrá un papel importante en el capítulo 5.

[†]Véase la sección 1.2.3.2
[6]Véase [14] pp. 261-263
[7]Véase [5].

Capítulo 4

Incompletud y consistencia

En este capítulo introducimos los conceptos de completud e incompletud de una teoría de primer orden, discutimos brevemente el primer teorema de incompletud de Gödel y demostramos el teorema de Chaitin.

4.1. Completud

El concepto de completud se aplica a las teorías axiomáticas de primer orden. Al respecto, daremos algunos ejemplos de teorías completas con el propósito de aclarar la definición dada.

Definición 4.1. Sea T una teoría axiomática de primer orden. Decimos que T es *completa* si para cualquier fórmula cerrada (*i.e.* que no tiene variables libres) B de L_T se cumple que $\vdash_T B$ ó $\vdash_T \neg B$.

Algunas teorías de primer orden completas son la de Tarski para la geometría euclidiana, la aritmética de Presburger y la teoría de campos algebraicamente cerrados con característica fija.

4.2. Primer teorema de incompletud de Gödel

En 1931 Gödel enunció sus dos teoremas de incompletud y demostró el primero de ellos en un artículo titulado *Sobre proposiciones formalmente*

indecidibles en Principia Mathematica y sistemas afines I [4]. En la demostración de su primer teorema utiliza un argumento basado en las paradojas de Richard y del mentiroso que a continuación bosquejamos y analizamos.

La demostración procede de la siguiente manera:

A través de una función inyectiva que a cada símbolo del lenguaje de la aritmética le asigna un número natural, Gödel codifica las fórmulas del lenguaje de la aritmética en números naturales. Posteriormente muestra que las propiedades y relaciones metateóricas entre las fórmulas y demostraciones pueden expresarse como propiedades y relaciones entre sus números de Gödel mediante la codificación anterior. Esto es de vital importancia, pues la codificación permite expresar a través de predicados recursivos propiedades importantes como "ser el número de Gödel de una prueba", "ser el número de Gödel de una prueba de la fórmula con número de Gödel x","ser un teorema de AR". Dichas propiedades se pueden codificar con las fórmulas $PRUEBA(x)$, $PR(x, y)$ y $TEO(x)$ que intuitivamente[1] se definen como sigue:

- $PRUEBA(x)$ si y sólo si x es el número de Gödel de una sucesión de fórmulas tal que cada fórmula de la sucesión es un axioma o puede deducirse a partir de las fórmulas anteriores a ella mediante alguna de las reglas de inferencia de AR.

- $PR(x, y)$ si y sólo si x es el número de Gödel de una prueba de la fórmula con número de Gödel y. Es decir, x satisface $PRUEBA(x)$ y satisface que y es el número de Gödel de su última fórmula.

- $TEO(x)$ si y sólo si existe y tal que $PR(y, x)$.

Una vez logrado esto, Gödel construye un enunciado G que indirectamente afirma ser indemostrable en la aritmética y demuestra que si la aritmética es consistente, entonces $\nvdash_{AR} G$. Del mismo modo, prueba que si la aritmética es ω-consistente, entonces $\nvdash_{AR} \neg G$.

De esta manera, Gödel no sólo demuestra la existencia de enunciados indecidibles, sino que también muestra explícitamente cómo construir uno de ellos para cada sistema que contenga una formalización de la aritmética recursiva.

Si bien la demostración que ofrece Gödel es bastante ingeniosa, también es demasiado compleja, pues requiere de un aparato teórico muy extenso y

[1] Para una definición formal de estas fórmulas, véase [18], pp. 124,125

de una fórmula cuya autorreferencia es confusa a primera vista. Esto motivó que matemáticos como Boolos y Chaitin dieran pruebas alternativas que, si bien no muestran explícitamente un enunciado indecidible, sí demuestran su existencia de un modo más simple. En la siguiente sección enunciamos y probamos el teorema de Chaitin, cuyo argumento utiliza complejidad Kolmogórov.

4.3. Complejidad e Incompletud: El teorema de Chaitin

En 1971 Chaitin publicó un artículo titulado *Complejidad computacional y el teorema de incompletud de Gödel* [2], en el cual muestra la existencia de enunciados aritméticos verdaderos e inderivables dentro de la aritmética de un modo sencillo. A continuación enunciamos y demostramos el teorema dado por Chaitin en dicho artículo.

En la demostración del teorema de Chaitin, será necesario utilizar lo siguiente:

Definición 4.2 (Condiciones de derivación de Hilbert y Bernays)**.** Una fórmula $P(x_0)$ cuya única variable libre es x_0 es un *predicado de prueba* para un sistema SF si para todos los enunciados A y B de L_{SF} se cumple lo siguiente:

D1. Si $\vdash_{SF} A$, entonces $\vdash_{SF} P(\ulcorner A \urcorner)$[†]

D2. $\vdash_{SF} P(\ulcorner A \urcorner) \rightarrow P(\ulcorner P(\ulcorner A \urcorner) \urcorner)$

D3. $\vdash_{SF} (P(\ulcorner A \rightarrow B \urcorner) \wedge P(\ulcorner A \urcorner)) \rightarrow P(\ulcorner B \urcorner)$

Lema 4.1. En particular, la fórmula $TEO(x_0) \equiv_{def} \exists x_1 \overline{PR}(x_1, x_0)$ es un predicado de prueba para AR.

Omitimos la demostración del lema 4.1 debido a su complejidad y extensión.[2]

Lema 4.2. Si AR es ω-consistente y $\vdash_{AR} TEO(\ulcorner \varphi \urcorner)$, entonces $\vdash_{AR} \varphi$.

[†]Véase Convenciones y notaciones.

[2]Para una demostración completa de este lema véase [18], capítulo 7.

Demostración. Supongamos que $\nvdash_{AR} \varphi$. En tal caso, los enunciados

$$\sim PR(0, \ulcorner\varphi\urcorner), \sim PR(1, \ulcorner\varphi\urcorner), \ldots, \sim PR(k, \ulcorner\varphi\urcorner), \text{ etc.}$$

son verdaderos. Como $\neg\overline{PR}(x_1, x_0)$ representa a esta relación en AR, tenemos que

$$\vdash_{AR} \neg\overline{PR}(\bar{0}, \ulcorner\varphi\urcorner), \vdash_{AR} \neg\overline{PR}(\bar{1}, \ulcorner\varphi\urcorner), \ldots, \vdash_{AR} \neg\overline{PR}(\bar{k}, \ulcorner\varphi\urcorner), \text{ etc.}$$

Por hipótesis, AR es ω-consistente, de modo que $\nvdash_{AR} \exists x_1 \overline{PR}(x_1, \ulcorner\varphi\urcorner)$, es decir, $\nvdash_{AR} TEO(\ulcorner\varphi\urcorner)$. Por contraposición en la metateoría, tenemos que si $\vdash_{AR} TEO(\ulcorner\varphi\urcorner)$, entonces $\vdash_{AR} \varphi$. □

Lema 4.3 (Teorema *s-m-n* de Kleene)**.** Sean $\{\varphi_i\}_{i\in\mathbb{N}}$ y $\{\varphi_i^{(2)}\}_{i\in\mathbb{N}}$ los conjuntos de funciones parcialmente computables de uno y dos argumentos respectivamente. Para toda $i \in \mathbb{N}$ y para toda $(x, y) \in Dom(\varphi_i^{(2)})$, existe una función total computable $s : \mathbb{N}^2 \to \mathbb{N}$ tal que $\varphi_{s(i,y)}(x) = \varphi_i^{(2)}(x, y)$.

Demostración. Consideremos la familia de funciones $\varphi_i^{(2)}(x, y)$ con y fijo y x, i variando sobre $\mathbb{N}$.

Fijando el valor de y, digamos $y = k$, podemos dar para cada $i \in \mathbb{N}$ una máquina de Turing $\mathcal{M}$ que computa $\varphi_i^{(2)}(x, k)$ como sigue:

Supongamos que T_j es una máquina de Turing tal que al iniciarse sobre una cinta en la que se encuentra escrito $\bar{x}$, mueve el cabezal hasta el final de $\bar{x}$, deja un espacio en blanco, imprime $\bar{k}$, regresa el cabezal a su posición inicial y se detiene. Entonces, $\mathcal{M} = T_i \circ T_j$ computa $\varphi_i^{(2)}(x, k)$ y su índice puede calcularse a partir de i, k. Es decir, existe una función total computable[1] $s : \mathbb{N}^2 \to \mathbb{N}$ tal que $\varphi_{s(i,k)}(x) = \varphi_i^{(2)}(x, k)$. □

Lema 4.4 (Teorema de la recursión)**.** Sean $\{T_k\}_{k\in\mathbb{N}}$ una enumeración de máquinas de Turing, $\{\varphi_k\}_{k\in\mathbb{N}}$ el conjunto de sus funciones asociadas[2] y $f : \mathbb{N} \to \mathbb{N}$ una función total computable. Entonces, existe $n \in \mathbb{N}$ tal que $T_{f(n)} \simeq T_n$. A dicho n se le conoce como *punto fijo* de f.

[1] La computabilidad de s es posible por la tesis de Church-Turing.

[2] Una función computable φ está asociada a una máquina de Turing T si sus dominios coinciden y $T(x)\downarrow\varphi(x)$ para todo x en su dominio.

Demostración. Consideremos la función $\psi : \mathbb{N}^2 \to \mathbb{N}$ definida como

$$\psi(x,y) = \begin{cases} f \circ \varphi_y(x) & \text{Si } \varphi_y(x)\downarrow \\ \text{No definida} & \text{Si } \varphi_y(x)\uparrow, \end{cases}$$

mediante la cual se podría hacer una diagonalización para obtener el índice buscado. Sin embargo, esto no es posible, ya que $\varphi_x(x)$ no necesariamente está definida para cada x.

Utilizando el lema 4.3, podemos construir una función $s : \mathbb{N} \to \mathbb{N}$ total computable a partir de $\psi(x,y)$ tal que $s(x) = f \circ \varphi_x(x)$ para cada $x \in \mathbb{N}$. Tomando $m \in \mathbb{N}$ tal que $\varphi_m = s$ e $i = \varphi_m(m)$, se tiene que $\varphi_i = \varphi_{\varphi_m(m)} = \varphi_{s(m)} = \varphi_{f\circ\varphi_m(m)} = \varphi_{f(i)}$. Es decir, $\varphi_i = \varphi_{f(i)}$.

Por lo tanto, $T_i \simeq T_{f(i)}$. □

Nota: Puesto que $K(x)$ se define mediante una minimalización, la existencia del correspondiente término dentro de AR podría no ser cierta. Para corregir eso, introduciremos la relación recursiva $C(x,y)$ definida como

$$C(x,y) \equiv_{def} (T_y(\varepsilon)\downarrow x \wedge \forall z(z < y \to \neg(T_z(\varepsilon)\downarrow x))),$$

que representa a la relación aritmética "$K(x) = y$" dentro de AR. Con base en ello, las relaciones "$K(x) > y$" y "$K(x) \leqslant y$" se representan respectivamente como "$C(x,z) \wedge z > y$" y "$\exists z(z \leqslant y \wedge C(x,z))$". Con la finalidad de hacer más cómoda la lectura de estas fórmulas, las abreviaremos como "$\bar{K}(x) = y$", "$\bar{K}(x) > y$" y "$\bar{K}(x) \leqslant y$" respectivamente.

Teorema 4.1 (de Chaitin)**.** Sea $K(y)$ la complejidad Kolmogórov de y. Si AR es ω-consistente, entonces existe una constante c tal que $\nvdash_{AR} \bar{K}(y) > c$ para todo número natural y.

Demostración. Sean $\{T_k\}_{k\in\mathbb{N}}$ una enumeración de máquinas de Turing, $m \in \mathbb{N}$ y T_n la máquina definida intuitivamente como sigue:

$T_n(\varepsilon)$ = "Encuentra el menor número natural x tal que x es (el número de Gödel de) una prueba de $\bar{K}(y) > m$ para algún número natural y, imprime dicho y sobre su cinta y se detiene."

Sea $f : \mathbb{N} \to \mathbb{N}$ una función recursiva total tal que $f(m) = n$, donde n y m son los mencionados anteriormente. Por el lema 4.4, existe $c \in \mathbb{N}$ tal que $T_c \simeq T_{f(c)}$ y tal que

$T_{f(c)}(\varepsilon)$ = "Encuentra el menor número natural x tal que x es (el número de Gödel de) una prueba de $\bar{K}(y) > c$ para algún número natural y, imprime dicho y sobre su cinta y se detiene."

Luego, como $T_c \simeq T_{f(c)}$, se tiene que

$T_c(\varepsilon)$ = "Encuentra el menor número natural x tal que x es (el número de Gödel de) una prueba de $\bar{K}(y) > c$ para algún número natural y, imprime dicho y sobre su cinta y se detiene."

Veamos que c es la constante buscada.

Supongamos que $T_c{\downarrow}$[‡]. Entonces, $\vdash_{AR} \exists x\overline{PR}(x, \ulcorner \bar{K}(y) > c \urcorner)$ por definición de T_c. Como por definición de TEO se cumple que $\exists x\overline{PR}(x, \ulcorner \bar{K}(y) > c \urcorner) \equiv TEO(\ulcorner \bar{K}(y) > c \urcorner)$, tenemos que $\vdash_{AR} TEO(\ulcorner \bar{K}(y) > c \urcorner)$. Luego, por la hipótesis de ω-consistencia y por el lema 4.2, $\vdash_{AR} \bar{K}(y) > c$. Sin embargo, $T_c{\downarrow}$ también implica por definición que $T_c{\downarrow}y$. Por lo tanto, $\vdash_{AR} \bar{K}(y) \leqslant c$.

Por lo anterior, $\vdash_{AR} (\bar{K}(y) > c) \wedge (\bar{K}(y) \leqslant c)$!

Por lo tanto, $T_c{\uparrow}$ es decir, $\vdash_{AR} \neg\exists x\overline{PR}(x, \ulcorner \bar{K}(y) > c \urcorner)$, o equivalentemente, $\vdash_{AR} \neg TEO(\ulcorner \bar{K}(y) > c \urcorner)$. Luego, por la hipótesis de que nuestra teoría es (ω-)consistente, sucede que $\nvdash_{AR} TEO(\ulcorner \bar{K}(y) > c \urcorner)$. Por la definición 4.2-D1, podemos concluir que $\nvdash_{AR} \bar{K}(y) > c$.

Por lo tanto, $\nvdash_{AR} \bar{K}(y) > c$ para todo número natural y. □

En este capítulo explicamos brevemente el primer teorema de Gödel, enunciamos y demostramos los teoremas *s-m-n* de Kleene y de la recursión, que formaron parte esencial de la demostración del teorema de Chaitin. En el siguiente capítulo explicaremos brevemente una demostración del segundo teorema de Gödel dada por Kikuchi y posteriormente demostraremos este teorema, utilizando una formalización del teorema de Chaitin.

[‡]Véase Convenciones y notaciones.

Capítulo 5

Segundo teorema de Gödel

En este capítulo damos una idea intuitiva de lo que el segundo teorema de Gödel dice, analizamos brevemente la demostración dada por Kikuchi en 1997 y, finalmente, hacemos una demostración más simple utilizando una formalización del teorema 4.1.

5.1. Idea intuitiva

Mientras que el primer teorema de Gödel afirma que si la aritmética es consistente, entonces es incompleta, el segundo afirma que en particular la aritmética es incapaz de probar el enunciado que afirma su consistencia. Tradicionalmente, la prueba de esto se basa en demostrar que $\vdash_{AR} CON \rightarrow G$, donde CON es un enunciado de L_{AR} que afirma la consistencia de AR y G es el enunciado de Gödel, pues si CON pudiera demostrarse dentro de la teoría, G también podría ser probado. Sin embargo, el primer teorema nos muestra que esto no es posible.

5.2. Demostración dada por Kikuchi

En 1997 Kikuchi publicó un artículo titulado *Kolmogorov complexity and the second incompleteness theorem* [8], en el cual da una nueva demostración del segundo teorema de Gödel, utilizando el teorema de Chaitin. Dicha demostración requiere diversos conceptos y teoremas previos que a continuación mencionamos y explicamos sin demostrar, pues sus demostraciones son

extensas y no son relevantes para el resto de esta tesis.

Definición 5.1 (Diagrama elemental)**.** Sea M una estructura correspondiente a un lenguaje L. Si $L(M)$ es una extensión de L que se obtiene al añadir a L un símbolo de constante c_a por cada elemento a de M, entonces M puede verse como una estructura correspondiente a $L(M)$ en la que los símbolos de L se interpretan de la misma manera que antes y cada símbolo de constante c_a se interpreta como su correspondiente elemento a. Al conjunto de enunciados de $L(M)$ que son verdaderos en M se le conoce como *diagrama elemental* de M.

Definición 5.2. Sean T una extensión recursivamente axiomatizable de AR y $\overline{L_{AR}} = L_{AR} \cup \mathcal{C}$, con $\mathcal{C}$ un conjunto de nuevos símbolos de constante. Decimos que una fórmula $\phi(x)$ de L_{AR} define un modelo de T en una teoría S si puede demostrarse dentro de S que el conjunto

$$\{\sigma : \sigma \text{ es un enunciado de } \overline{L_{AR}} \text{ que satisface } \phi(\ulcorner\sigma\urcorner)\}$$

forma un diagrama elemental de un modelo de T con dominio $\mathcal{C}$.

Teorema 5.1. Existe una fórmula $TR_T(x)$ en L_{AR} que define un modelo de T en $AR + CON(T)$.

Definición 5.3. Sean $\mathfrak{M}$ y $\mathfrak{M}'$ estructuras de L_{AR}. Decimos que $\mathfrak{M}'$ es una *extensión final* de $\mathfrak{M}$ ($\mathfrak{M} \subseteq_e \mathfrak{M}'$) si $\mathfrak{M} \subseteq \mathfrak{M}'$ y $\mathfrak{M}' \vDash a < b$ para todo $a \in \mathfrak{M}$ y para todo $b \in \mathfrak{M}' \backslash \mathfrak{M}$.

Definición 5.4. Sean $\mathfrak{M}$ un modelo de AR y $\mathfrak{M}'$ un modelo de T. Decimos que $\mathfrak{M}'$ es una extensión final definible de $\mathfrak{M}$ ($\mathfrak{M} \subseteq_d \mathfrak{M}'$) si $\mathfrak{M} \subseteq_e \mathfrak{M}'$ y

$$\mathfrak{M} \vDash TEO(\ulcorner\phi\urcorner) \Rightarrow \mathfrak{M}' \vDash \phi,$$

$$\mathfrak{M} \vDash TR_T(\ulcorner\phi\urcorner) \Leftrightarrow \mathfrak{M}' \vDash \phi$$

para alguna fórmula $TR_T(x)$ de L_{AR}.

Corolario 5.1. Sea $\mathfrak{M}$ un modelo de AR. Entonces, $\mathfrak{M}$ satisface $CON(T)$ si y sólo si $\mathfrak{M}$ tiene una extensión final definible que es modelo de T.

Lema 5.1. Para todo número natural a, existe $b \leqslant a+1$ tal que $a+1 \leqslant K(b)$.

Lema 5.2. $\vdash_T \forall x \exists y (y \leqslant x+1 \wedge x+1 \leqslant \bar{K}(y))$.

Utilizando lo anterior y una formalización del teorema 4.1, que demostraremos más adelante (proposición 5.1), podemos proceder con la demostración dada por Kikuchi.

La idea intuitiva detrás de su demostración es tomar un modelo $\mathfrak{M}_0$ de la aritmética junto con un número $a_0 \leqslant c+1$, donde c es la constante del teorema 4.1, que satisface ser el primer número con complejidad Kolmogórov mayor que c. Puesto que esta propiedad no puede demostrase en AR, su negación puede añadirse como axioma consistentemente, por lo que $\mathfrak{M}_0$ también satisface el enunciado que afirma la consistencia de la nueva teoría $T + \bar{K}(a_0) \leqslant c$.

Luego, por el corolario 5.1, podemos asegurar la existencia de $\mathfrak{M}_1$, una extensión final definible de $\mathfrak{M}_0$. Si nuevamente tomamos un elemento $a_1 \leqslant c+1$ que satisface ser el primero con complejidad Kolmogórov mayor que c, podemos repetir el proceso anterior y dar una extensión final definible $\mathfrak{M}_2$ de $\mathfrak{M}_1$. Además, podemos asegurar que $a_0 < a_1$, pues al ser $\mathfrak{M}_1$ una extensión final definible de $\mathfrak{M}_0$, se cumple que $\mathfrak{M}_1 \vDash a < b$ para todo $a \in \mathfrak{M}_0$ y para todo $b \in \mathfrak{M}_1 \backslash \mathfrak{M}_0$.

Si repetimos los procedimientos anteriores c veces más, obtendremos una sucesión $\mathfrak{M}_0 \subseteq_d \mathfrak{M}_1 \subseteq_d \ldots \subseteq_d \mathfrak{M}_{c+2}$ de modelos de la aritmética y una correspondiente sucesión $a_0 < a_1 < \ldots < a_{c+2}$ de elementos estrictamente creciente. Sin embargo, esto no es posible, pues habríamos encontrado $c+2$ números menores que $c+1$. Por lo tanto, concluimos que debe existir al menos un modelo de la aritmética que no satisface el enunciado que afirma su consistencia, y por el teorema de completud de Gödel, dicho enunciado no puede ser demostrable dentro de la aritmética. A continuación hacemos un bosquejo de la demostración formal.

Teorema 5.2. Si T es consistente, $CON(T)$ no es demostrable en T.

Demostración. Supongamos que T es consistente y que cualquier modelo de T satisface $CON(T)$. Puesto que T es consistente, podemos asegurar que tiene al menos un modelo, digamos $\mathfrak{M}_0$. Por el lema 5.2 y por el principio del buen orden, existe $a_0 \leqslant c+1$[†] tal que

$$\mathfrak{M}_0 \vDash c < \bar{K}(a_0) \wedge \forall x(x < a_0 \rightarrow \bar{K}(x) \leqslant c).$$

Por la proposición 5.1, $\mathfrak{M}_0 \vDash \neg TEO(\ulcorner \bar{K}(a_0) > c \urcorner)$.

[†] c es la constante del teorema 4.1.

Por lo tanto, $\mathfrak{M}_0 \vDash CON(T + \bar{K}(a_0) \leqslant c)$. Luego, por el corolario 5.1, $\mathfrak{M}_0$ tiene una extensión final definible $\mathfrak{M}_1$. Ahora, si nuevamente tomamos el menor elemento $a_1 \leqslant c+1$ tal que $\mathfrak{M}_1 \vDash c < \bar{K}(a_1)$, tenemos que $\mathfrak{M}_1 \vDash \bar{K}(a) \leqslant c$ para todo $a < a_0$, y $\mathfrak{M}_1 \vDash \bar{K}(a_0) \leqslant c$, pues $\mathfrak{M}_1$ es modelo de $T + \bar{K}(a_0) \leqslant c$. De este modo, sucede que $\mathfrak{M}_1 \vDash \forall x(x \leqslant a_0 \to \bar{K}(x) \leqslant c)$, por lo que necesariamente $a_0 < a_1$.

Si repetimos lo anterior $c+1$ veces más, tendremos una sucesión $\mathfrak{M}_0 \subseteq_d \mathfrak{M}_1 \subseteq_d \ldots \subseteq_d \mathfrak{M}_{c+2}$ de modelos de T y una sucesión estrictamente creciente de números $a_0 < a_1 < \ldots < a_{c+2}$. Sin embargo, esto no es posible, pues cada a_i debe ser menor o igual que $c+1$.

Por lo tanto, si T es consistente, existe un modelo de T que no satisface $CON(T)$. Luego, por el teorema de completud de Gödel, tenemos que $CON(T)$ no es demostrable en T. □

5.3. Demostración del segundo teorema de Gödel

En 2010 Kritchman y Raz publicaron un artículo titulado *The surprise examination paradox and the second incompleteness theorem* [10], en el cual dan una demostración del segundo teorema de Gödel a partir del teorema de Chaitin. Como vimos en la sección anterior, esto ya había sido realizado por Kikuchi en 1997. No obstante, la prueba de Kritchman y Raz es mucho más sencilla, pues no recurre a la teoría de modelos. En esta sección daremos una prueba del segundo teorema de Gödel basada en la de Kritchman y Raz.

Antes de demostrar el segundo teorema de Gödel necesitamos probar la siguiente proposición:

Proposición 5.1 (Formalización del teorema 4.1)**.**

$$\vdash_{AR} CON \to \forall x \neg TEO(\ulcorner \bar{K}(x) > c \urcorner)$$

Demostración. Procedamos por contraposición en AR.

Consideremos el predicado extendido de Kleene $T_1(x, y, z, w)$, el cual es recursivo y cuya definición es

$$T_1(x, y, z, w) = \begin{cases} 1 & \text{si } M_x(y)\!\downarrow\! w \text{ en } z \text{ pasos,} \\ 0 & \text{en cualquier otro caso.} \end{cases}$$

Tomemos el enunciado de consistencia $CON \equiv_{def} \neg TEO(\ulcorner \varphi \wedge \neg \varphi \urcorner)$ (con φ cualquier fórmula de L_{AR}) y una función recursiva total $f : \mathbb{N} \to \mathbb{N}$ con punto fijo[1] c tal que

$$\exists x, n[T_1(f(c), \varepsilon, x, n) = 1]$$

y

$$\exists y < x[T_1(f(c), \varepsilon, y, \mu k\overline{PR}(k, \ulcorner \bar{K}(n) > c \urcorner)) = 1].$$

La lectura metateórica del primer enunciado afirma que existen dos números naturales x, n tales que $M_{f(c)}(\varepsilon)$ converge a n en x pasos. La lectura metateórica del segundo enunciado afirma que existe un número natural $y < x$ tal que $M_{f(c)}(\varepsilon)$ converge al menor número natural k tal que k es el número de Gödel de una prueba del enunciado $\bar{K}(n) > c$, en y pasos.

Por el lema 4.4, $M_{f(c)} \simeq M_c$, por lo que

$$\exists x, n[T_1(c, \varepsilon, x, n) = 1]$$

y

$$\exists y < x[T_1(c, \varepsilon, y, \mu k\overline{PR}(k, \ulcorner \bar{K}(n) > c \urcorner)) = 1],$$

cuya lectura metateórica afirma que existen dos números naturales x, n tales que $M_c(\varepsilon)$ converge a n en x pasos, y que existe un número natural $y < x$ tal que $M_c(\varepsilon)$ converge al menor número natural k tal que k es el número de Gödel de una prueba del enunciado $\bar{K}(n) > c$, en y pasos. Es decir, afirman la existencia de la máquina utilizada en la demostración del teorema 4.1.

Puesto que estos enunciados son teoremas de AR si y sólo si el enunciado $\exists n[TEO(\ulcorner \bar{K}(n) > c \urcorner)]$ es teorema de AR, supongamos que $\vdash_{AR} \exists n[TEO(\ulcorner \bar{K}(n) > c \urcorner)]$. Entonces, en AR podemos probar lo siguiente:

[1] Véase el lema 4.4.

1.	$\vdash_{AR} \exists n[TEO(\ulcorner \bar{K}(n) > c \urcorner)]$	Hip.
2.	$\vdash_{AR} \exists n[TEO(\ulcorner \bar{K}(n) > c \urcorner)] \rightarrow M_c\downarrow$	Def. de M_c
3.	$\vdash_{AR} M_c\downarrow$	M.P 1,2.
4.	$\vdash_{AR} \exists x, n[T_1(c, \varepsilon, x, n) = 1]$	Consec. de 3.
5.	$\vdash_{AR} \bar{K}(n) \leqslant c$	Por 4 y def. de $\bar{K}(x)$.
6.	$\vdash_{AR} TEO(\ulcorner \bar{K}(n) \leqslant c \urcorner)$	Def. 4.2-D1 a 5.
7.	$\vdash_{AR} \exists y < x[T_1(c, \varepsilon, y, \mu k \overline{PR}(k, \ulcorner \bar{K}(n) > c \urcorner) = 1]$	Consec. de 3.
8.	$\vdash_{AR} \exists k \overline{PR}(k, \ulcorner \bar{K}(n) > c \urcorner)$	Consec. de 7.
9.	$\vdash_{AR} \exists j \overline{PR}(j, \ulcorner \bar{K}(n) \leqslant c \urcorner)$	Def. de TEO a 6.
10.	$\vdash_{AR} \overline{PR}(k_0, \ulcorner \bar{K}(n) > c \urcorner)$	Regla C a 8.
11.	$\vdash_{AR} \overline{PR}(j_0, \ulcorner \bar{K}(n) \leqslant c \urcorner)$	Regla C. a 9.
12.	$\vdash_{AR} PRUEBA(k_0)$	Def. de PR a 10.
13.	$\vdash_{AR} PRUEBA(j_0)$	Def. de PR a 11.
14.	$\vdash_{AR} PRUEBA(k_0) \wedge PRUEBA(j_0)$	Conj. 12 y 13.
15.	$\vdash_{AR} (PRUEBA(k_0) \wedge PRUEBA(j_0)) \rightarrow PRUEBA(k_0 * j_0)$	Véase [18] pp. 214-217.
16.	$\vdash_{AR} PRUEBA(k_0 * j_0)$	M.P. 14 y 15.
17.	$\vdash_{AR} PRUEBA(k_0 * j_0 * \ulcorner \bar{K}(n) > c \wedge \bar{K}(n) \leqslant c \urcorner)$	Def. de $PRUEBA$.
18.	$\vdash_{AR} \overline{PR}(m_0, \ulcorner \bar{K}(n) > c \wedge \bar{K}(n) \leqslant c \urcorner)$[3]	Def. de PR a 17.
19.	$\vdash_{AR} \exists m \overline{PR}(m, \ulcorner \bar{K}(n) > c \wedge \bar{K}(n) \leqslant c \urcorner)$	Regla $E4$ a 18.
20.	$\vdash_{AR} TEO(\ulcorner \bar{K}(n) > c \wedge \bar{K}(n) \leqslant c \urcorner)$	Def. de TEO a 19.
21.	$\vdash_{AR} \neg CON$	Equiv. a 20.
22.	$\vdash_{AR} \exists n[TEO(\ulcorner \bar{K}(n) > c \urcorner)] \rightarrow \neg CON$	T.D. 1-21.

Por contraposición en AR, $\vdash_{AR} CON \rightarrow \forall n \neg TEO(\ulcorner \bar{K}(n) > c \urcorner)$. □

Teorema 5.3 (segundo teorema de Gödel)**.** Si AR es consistente, entonces $\nvdash_{AR} CON$.

Demostración. Consideremos $\{T_i\}_{i \in \mathbb{N}}$ una enumeración de máquinas de Turing con 3 símbolos. Notemos que sólo hay $(6n)^{2n}$ máquinas de Turing con 3 símbolos y n estados[3].

Por la proposición 5.1, sabemos que $\vdash_{AR} CON \rightarrow \forall x \neg TEO(\ulcorner \bar{K}(x) > c \urcorner)$. Por lo que en particular sucede que

[2] $m_0 = k_0 * j_0 * \ulcorner \bar{K}(n) > c \wedge \bar{K}(n) \leqslant c \urcorner$.

[3] Esto es debido a que hay 6 posibles movimientos: moverse a la derecha, moverse a la izquierda, escribir blanco, escribir 0, escribir 1 y detenerse.

$$\vdash_{AR} CON \rightarrow \forall x[x \leqslant (6c)^{2c} \rightarrow \neg TEO(\ulcorner \bar{K}(x) > c \urcorner)] \quad (*).$$

Sea $m = Card(\{x \in \mathbb{N} : x \leqslant (6c)^{2c} \wedge \bar{K}(x) > c\})$[‡].

Veamos (por inducción) que si $\vdash_{AR} CON$, entonces $\vdash_{AR} (m \geqslant i+1)$ para todo $i \leqslant (6c)^{2c} + 1$ (lo cual contradecirá su definición).

Caso base: Como hay $(6c)^{2c}$ máquinas de Turing con 2 símbolos y c estados, existe al menos un número $x \leqslant (6c)^{2c}$ tal que $\bar{K}(x) > c$, por lo que $m \geqslant 1$.

Supongamos que $m = 1$, entonces existe un único $x \leqslant (6c)^{2c}$ tal que $\bar{K}(x) > c$ y cualquier otro $y \leqslant (6c)^{2c}$ cumple que $\bar{K}(y) \leqslant c$.

Como $\vdash_{AR} \bar{K}(y) \leqslant c$ y $\vdash_{AR} m \geqslant 1$, el x para el cual no se prueba que $\bar{K}(x) \leqslant c$, debe cumplir que $\bar{K}(x) > c$. Es decir, si $m = 1$, entonces existe x tal que $\vdash_{AR} \bar{K}(x) > c$, lo cual no puede suceder si AR es consistente.

Por lo tanto, $m > 1$, es decir, $m \geqslant 2$.

Hipótesis de inducción: Supongamos que para toda $1 \leqslant x \leqslant (6c)^{2c} + 1$, con $x \leqslant i$ se cumple que $\vdash_{AR} (m \geqslant i)$.

Paso inductivo: Sea $x = i + 1$. Por demostrar, $\vdash_{AR} (m \geqslant i + 1)$.

Sea $r = (6c)^{2c} - i$.

Por definición de m, $\vdash_{AR} (m = i) \rightarrow \exists_d y_1, \ldots, y_r \leqslant (6c)^{2c}[(\bar{K}(y_1) \leqslant c) \wedge \ldots \wedge (\bar{K}(y_r) \leqslant c)]$.[4]

Por la definición 4.2 y por el hecho de que $TEO(x)$ es un predicado de prueba para AR, sabemos que $\vdash_{AR} \forall y[y \leqslant (6c)^{2c} \rightarrow (\bar{K}(y) \leqslant c \rightarrow TEO(\ulcorner \bar{K}(y) \leqslant c \urcorner))]$.

[‡]Véase Convenciones y notaciones.

[4] $\exists_d x_1, \ldots, x_n \leqslant k P(x_1, \ldots, x_n) \equiv_{def.} \exists x_1, \ldots, x_n([\forall i, j(1 \leqslant i, j \leqslant n \rightarrow (i \neq j \rightarrow x_i \neq x_j))] \wedge [(\forall m(1 \leqslant m \leqslant n \rightarrow x_m \leqslant k) \wedge P(x_1, \ldots, x_n)])$.

Luego, $\vdash_{AR} (m = i) \to \exists_d y_1, \ldots, y_r \leqslant (6c)^{2c}[TEO(\ulcorner \bar{K}(y_1) \leqslant c \urcorner) \wedge \ldots \wedge TEO(\ulcorner \bar{K}(y_r) \leqslant c \urcorner)]$.

Entonces, para cada $y_1, \ldots, y_r \leqslant (6c)^{2c}$ distintos entre sí y para todo $x \leqslant (6c)^{2c}$ tal que $x \neq y_1, \ldots, y_r$, se tiene que $\vdash_{AR} (m \geqslant i) \to [((\bar{K}(y_1) \leqslant c) \wedge \ldots \wedge (\bar{K}(y_r) \leqslant c)) \to (\bar{K}(x) > c)]$.

Luego, por la definición 4.2, $\vdash_{AR} TEO(\ulcorner m \geqslant i \urcorner) \to ([TEO(\ulcorner \bar{K}(y_1) \leqslant c \urcorner) \wedge \ldots \wedge TEO(\ulcorner \bar{K}(y_r) \leqslant c \urcorner)] \to TEO(\ulcorner \bar{K}(x) > c \urcorner))$.

Por lo anterior, $\vdash_{AR} (m = i \wedge TEO(\ulcorner m \geqslant i \urcorner)) \to \exists x[x \leqslant (6c)^{2c} \wedge TEO(\ulcorner \bar{K}(x) > c \urcorner)]$, que es equivalente a $\vdash_{AR} TEO(\ulcorner m \geqslant i \urcorner) \to [(m = i) \to \exists x(x \leqslant (6c)^{2c} \wedge TEO(\ulcorner \bar{K}(x) > c \urcorner))]$.[6]

Como por hipótesis de inducción $\vdash_{AR} (m \geqslant i)$, $\vdash_{AR} TEO(\ulcorner m \geqslant i \urcorner)$ por la definición 4.2-D1.

Por lo que $\vdash_{AR} (m = i) \to \exists x[x \leqslant (6c)^{2c} \wedge TEO(\ulcorner \bar{K}(x) > c \urcorner)]$.

Por contraposición, $\vdash_{AR} \forall x[x \leqslant (6c)^{2c} \to \neg TEO(\ulcorner \bar{K}(x) > c \urcorner)] \to \neg(m = i)$.

Luego, por $(*)$ y transitividad de la implicación[7], $\vdash_{AR} CON \to \neg(m = i)$.

Como por hipótesis $\vdash_{AR} CON$, $\vdash_{AR} \neg(m = i)$.

Por lo anterior y por la hipótesis de inducción[8], $\vdash_{AR} (m \geqslant i + 1)$.

En particular, $\vdash_{AR} m > (6c)^{2c}$. Como $\vdash_{AR} m \leqslant (6c)^{2c}$ por definición de m, entonces $\vdash_{AR} [m > (6c)^{2c}] \wedge [m \leqslant (6c)^{2c}]$!

Por lo tanto, $\nvdash_{AR} CON$.

□

[6] Pues la conjunción es conmutativa y $((A \wedge B) \to C) \equiv (A \to (B \to C))$.
[7] $(P \to Q), (Q \to R) \vdash (P \to R)$.
[8] Pues $x \geqslant y \equiv (x > y \vee x = y)$, $x > y \equiv x \geqslant y + 1$ y $(P \vee Q), \neg Q \vdash P$.

Capítulo 6

Conclusiones

Para concluir este trabajo, haremos una breve recapitulación del mismo y destacaremos algunos puntos.

En el primer capítulo definimos y explicamos los conceptos de función recursiva, función recursiva primitiva, máquina de Turing y máquina de Turing universal, que tuvieron un papel esencial en las definiciones y demostraciones de los capítulos 3, 4 y 5.

En el segundo capítulo comenzamos introduciendo algunas definiciones y resultados propios de la teoría de modelos y de la teoría de ordinales, que posteriormente nos ayudaron a definir y comparar los conceptos de consistencia, ω-consistencia y completud.

En el tercer capítulo definimos el concepto de complejidad Kolmogórov, hicimos notar que la complejidad de un número puede variar considerablemente dependiendo de la definición utilizada y mencionamos algunas de sus aplicaciones.

En el cuarto capítulo explicamos brevemente el primer teorema de Gödel, demostramos el teorema s-m-n de Kleene, el teorema de la recursión y el teorema de Chaitin, cuya formalización tuvo gran importancia en el capítulo 5.

En el quinto capítulo dimos una explicación intuitiva de la demostración

del segundo teorema de Gödel hecha por Kikuchi y presentamos una demostración más simple utilizando una formalización del teorema de Chaitin.

Ante la existencia de múltiples demostraciones previas del segundo teorema de Gödel, nos parece prudente resaltar la importancia de la que presentamos:

- A diferencia de muchas demostraciones existentes, la nuestra es meramente sintáctica.
- El argumento es más claro desde un punto de vista intuitivo y más fácil de entender. En términos simples, hemos demostrado que si la aritmética fuera capaz de probar su propia consistencia, sería posible encontrar dos números naturales a y b tales que $a < b$ y $a \geqslant b$, lo cual claramente es imposible.
- Al utilizar como herramienta principal el teorema de Chaitin, nuestra demostración muestra claramente que hay una fuerte y poco evidente relación entre la lógica matemática y la teoría matemática de la información, las cuales, al ser parte de las matemáticas puras y aplicadas respectivamente, a simple vista pueden parecer muy distantes.
- La relación mencionada en el punto anterior es poco evidente, ya que se debe hacer una triangulación a través de la teoría matemática de la computación, mediante el teorema *s-m-n* de Kleene (lema 4.3) y mediante el teorema de la recursión (lema 4.4) para notarla.

Bibliografía

[1] Bridges, D.S. (1994). *Computability. A Mathematical sketchbook.* Hamilton: Springer-Verlag.

[2] Chaitin, G.J. (1971). Computational Complexity and Gödel's Incompleteness Theorem. *ACM SIGACT News*, 9, 11-12.

[3] Fernández de Castro, M., Villegas, L.M. (2011). *Lógica Matemática II: Clásica, Intuicionista y Modal.* México: Universidad Autónoma Metropolitana.

[4] Gödel, K., (1931). Über formal unentscheidbare Sätze der *Principia Mathematica* und verwandter Systeme. I. *Monatshefte für Mathematik und Physik.*, 38, 173-198. [tr. al español, en [13], 53-87]

[5] Hernandez-Orallo, J. (1999). Beyond the Turing Test. *Journal of Logic, Language and Information.*, 9, 1-20.

[6] Hrbacek, K., Jech, T. (1999) *Introduction to Set Theory.* Nueva York: Marcel Dekker, Inc.

[7] Kikuchui, M. (1994). A Note On Boolos' Proof of the Incompleteness Theorem. *Math. Logic Quart.*, 40, 528-532.

[8] Kikuchi, M. (1997). Kolmogorov Complexity and the second incompleteness theorem. *Arch. Math. Logic*, 36, 437-443.

[9] Kolmogorov, A. N. (1968). Three approaches to the quantitative definition of information. *International Journal of Computer Mathematics*, 2, 157-168.

[10] Kritchman, S., Raz, R. (2010). The Surprise Examination Paradox and the Second Incompleteness Theorem. *Notices of the AMS*, 57(11), 1454-1458.

[11] Mendelson, E. (2015). *Introduction to mathematical logic.* (6ª ed.) Nueva York: CRC Press.

[12] Miranda, F. E., Rojas, D., Villegas, L. M. (2000). *Conjuntos y modelos. Curso avanzado*. México: Universidad Autónoma Metropolitana.

[13] Mosterín, J., (2006). *Kurt Gödel, Obras Completas.* (2ª ed.) Madrid: Alianza Editorial.

[14] Odifreddi, P. (1989). *Classical Recursion Theory. The Theory of Functions and Sets of Natural Numbers*. Amsterdam: Elsevier Science.

[15] Petzold, C. (2008). *The Annotated Turing. A Guided Tour through Alan Turing's Historic Paper on Computability and the Turing Machine.* Indianapolis: Wiley Publishing Inc.

[16] Post, E. (1947). Recurisve unsolvability of a problem of thue. *Journal of Symbolic Logic*, 12, 1-11.

[17] Raatikainen, P. (1998). On interpreting Chaitin's incompleteness theorem. *Journal of Philosophical Logic*, 27, 569-586.

[18] Torres, C. *Limitaciones internas de la aritmética formalizada. Un estudio con especial énfasis en el segundo teorema de Gödel.* México. No publicado.

[19] Turing, A. M. (1936). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42(2), 230-265.