Universidad Michoacana de San Nicolás de Hidalgo

Facultad de Ciencias Físico-Matemáticas

Mat. Luis Manuel Rivera Gutiérrez

Problemas de Clasificación Algorítmica

Tesis

Que para obtener el título de:
Licenciado en Ciencias Físico-Matemáticas

Presenta:
Daniel Raggi Pérez

Asesor:
Dr. David Meza Alcántara

Morelia, Michoacán, México.
Diciembre, 2009

Agradecimientos

Tengo mucho que agradecer, pero dificultades para hacerlo. De todas maneras, aquí va mi mejor intento:

- Les agradezco mucho a Gerardo, Malú y Miguel; a ustedes les debo lo más profundo e importante de mi formación. Y a Tanja porque, donde sea y cuando sea, tú eres la misma Tanja empática y compasiva.
- A David, mi asesor y amigo, porque me has dado muchísimo; quizá más de lo que te imaginas. Me ayudaste a buscar el tema perfecto. Me guiaste, me tuviste paciencia, y siempre tuve toda tu atención cuando la necesité.
- A mis sinodales y maestros: Fernando, Mota, Luis, Rigo, Salvador, Malú, David, Gerardo y Jorge.
- A mis amigos de la Facultad y asociados: Tere, Poke, Mike, Abdón, Tania, Nazi, Ana, Héctor, Julián, Fabi, Ahtziri, Juan Pablo y Osvaldo; porque sé que ahí están.
- A los amigos que veo muy poco, pero que todavía representan algo importante para mí: Vicente, Marevna, Uriel, Nadia B., Nadia R., Emilio, Rocío, Andrea J. y Lubianca.
- A mis familiares y asociados, que tanto disfruto cada vez que los vuelvo a ver: Luciana, Amanda, Héctor, Javier, Carmen, Gerardo, Janet, Aldo, Enrique, Mauricio y Vicky.
- A otro tipo de amigos: Remy, Lupita, et al., Mireya, Luis, et al., Olinda et al., Edgar, Luis Miguel y Maria Elena.
- A Edgar Acosta, porque me diste una pequeña idea que aparece en la tesis, y una gran idea que no aparece en la tesis.
- A Diego Rojas, porque te interesaste en el tema y promoviste el razonamiento. A Carlos Torres y a Douglas Hofstadter porque, aunque no se enteraron, ustedes comenzaron este trabajo; al menos introdujeron la semillita mental de que Gödel tiene que ver con computabilidad.
- De nuevo, a Malú; porque además de todo, me enseñaste muchísimas matemáticas.

- De nuevo, a Fernando; porque contigo aprendí una forma de ver las matemáticas, pero más que nada, te agradezco por tener abierto ese corazonzote.
- De nuevo, a Fabi; por aquéllo.
- A Kima, porque nos dedicó toda su vida y a O'Kuri, porque está trabajando en eso. A Medo y a Gole, por los pequeños momentos.

Y, por si falta alguien, agradeceré a unos arquetipos tales que, como en los horóscopos, uno no puede evitar identificarse con alguno de ellos. Son las personificaciones de tres conceptos que se aparecen siempre y donde sea:

1. Al *incomprensible* que, en su oscura sabiduría, se mantiene consistente y completo; arrojando nuevos datos -información infalible- durante cada crisis.
2. Al *inconsistente* que, en su confusa libertad, se mantiene comprensible y completo; entendiendo el verdadero valor de todo esto: nada.
3. Al *incompleto* que, en gran humildad y ambición, se mantiene comprensible y consistente; y nunca para de buscar la verdad.

Índice general

Introducción

En 1910 Bertrand Russell y Alfred North Whitehead publicaron los libros *Principia Mathematica*, donde se expone una axiomatización de la teoría de conjuntos, con la tesis de que de ésta se derivan todas las verdades matemáticas. En 1931 Kurt Gödel probó una limitación inevitable en las teorías formales que incluía a aquella expuesta en *Principia Mathematica*.

Por otro lado, en 1930, Alonzo Church introdujo el Cálculo Lambda y en 1936, Alan Turing introdujo el concepto de Máquina de Turing; conceptos relativamente simples y equivalentes, que formalizan a la noción de *algoritmo*. Al rededor de 1936 ambos demostraron, de manera independiente, una limitación para sus respectivas formalizaciones este concepto (que, de hecho, se hereda a las teorías formales, por la naturaleza algorítmica de éstas).

En 1953 Henry Gordon Rice demostró un resultado que representa una limitación más fuerte para los algoritmos. Su teorema muestra una problemática que se aparece sobre cualquier clasificación de los algoritmos.

Las limitaciones en computabilidad y las limitaciones en las teorías formales tienen una naturaleza muy parecida, y esto quizá no es muy sorprendente dado que Turing se inspiró en la prueba de Gödel para probar la primera limitación, en 1936.

En esta tesis se pretende analizar ese paralelismo, y llevarlo más allá, extrayendo la esencia del teorema de Rice para formar un resultado que muestra una limitación análoga en la clasificación de las oraciones de una teoría formal. Lamentablemente, el paralelismo no es perfecto y el análogo a un lema, que se usa para probar el teorema de Rice, no es cierto para las teorías formales, por lo que la prueba no se puede heredar con un cambio de nombre, como en varias otras ocasiones que sí se puede hacer. Sin embargo, sí se encontró una prueba para el resultado análogo al teorema de Rice y ésta sí resultó ser heredable para el teorema de Rice. Estas dos son las pruebas que se expondrán en el último capítulo; no se expondrá la demostración usual del teorema de Rice.

Para entender el material de la tesis no se necesita más que las matemáticas más básicas de una licenciatura en matemáticas. Quizá se necesita saber un poco de lógica de proposiciones y sería ventajoso tener familiaridad con la forma peculiar con la que se trabaja la lógica.

Capítulo 1

Los Objetos

1.1. Máquinas de Turing

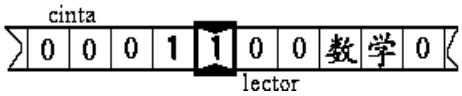
1.1.1. Como operadores de símbolos

Inexistentes físicamente (porque hasta donde yo sé nadie ha construido una, aunque no me sorprendería que alguien ya lo hubiera hecho, nada más por diversión) las *Máquinas de Turing* (MT) son una formalización de la noción de *algoritmo* basada en la descripción de operaciones simbólicas, como la de sustituir todas las apariciones de un símbolo por otro en un texto, la de borrar un segmento de éste, duplicarlo, generar un nuevo texto que sea una crítica literaria del primero, etc. Dado el poder intrínseco en la definición, sería interesante que, después de entender el funcionamiento de las MT, el lector intentara imaginar una operación simbólica no realizable por una de éstas. Difícil empresa sería, pues resulta que las Máquinas de Turing son una buena representación de lo que las computadoras modernas pueden hacer.

Imaginemos a una MT como una *cinta* infinita hacia ambos lados (piénsese como "suficientemente grande" si hay conflicto con la posibilidad de una construcción física de alguna), partida discretamente en *celdas* en las que se puede escribir símbolos (uno por cada celda), con un *lector* que lee una sola celda a la vez y reacciona a los diferentes estímulos cambiando el símbolo de la celda, moviéndose a alguna celda adyacente, o cambiando la forma en la que ella misma reaccionará al siguiente estímulo. Para esto, necesitamos un *alfabeto* A, el cual es un conjunto finito con al menos un símbolo 0, un *conjunto de estados* Q de los cuáles depende la reacción al estímulo. Éste conjunto finito deberá tener al menos dos estados llamados *estado inicial* (q_0) y *estado final* (q_f) que representan el estado en el que debe estar la máquina cuando em-

pieza y termina un trabajo, respectivamente, y finalmente necesitamos definir una *función de transición* $\delta\colon A \times Q \setminus \{q_f\} \to A \times Q \times \{\leftarrow, \rightarrow\}$ (este último conjunto representa los movimientos; izquierda y derecha), que diga cómo es que la máquina reaccionará a lo que lee; por ejemplo, si $\delta(1, q_0) = (0, q_1, \rightarrow)$, eso quiere decir que si la máquina está en el estado inicial con el lector en una celda que tiene escrito el símbolo 1, la máquina procederá a escribir el símbolo 0 en lugar de 1, cambiará del estado q_0 al estado q_1, y luego se moverá a la derecha. Cabe mencionar que, como el estado final representa la finalización de un cómputo, lo que queda escrito en la cinta puede interpretarse como el resultado de aplicar el *algoritmo* (al cual la máquina representa) a lo que estaba escrito en la cinta antes de comenzar.

Una vez que se dan los dos conjuntos y la función, queda descrita una máquina de Turing.



Para familiarizarnos un poco con el uso de las MT, daré algunos ejemplos de máquinas con explicación de lo que hacen.

1. Sean $A = \{0, 1\}$ y $Q = \{q_0, q_1, q_f\}$.
   Sea δ función en la que $\delta(1, q_0) = (0, q_1, \rightarrow)$, $\delta(0, q_0) = (1, q_1, \rightarrow)$, $\delta(1, q_1) = (0, q_f, \Lambda)$ y $\delta(0, q_1) = (1, q_f, \rightarrow)$.
   Lo que hace esta máquina es cambiar el símbolo de la celda en la que empieza, así como el de la celda a su derecha inmediata y luego se detiene.

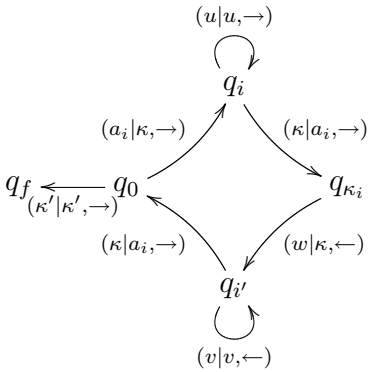
2. Sean $A = \{0, 1\}$ y $Q = \{q_0, q_f\}$.
   Sea δ función en la que $\delta(1, q_0) = (0, q_0, \rightarrow)$ y $\delta(0, q_0) = (0, q_0, \rightarrow)$.
   Esta máquina cambia todos los 1's por 0's a la derecha de donde comienza el lector, incluida la celda donde comienza, y nunca se detiene. En general, decimos que una máquina *no realiza un cómputo* con cierta entrada, cuando alimentada con dicha entrada, la máquina no llega al estado final (llamaremos *entrada* a lo que está escrito en la cinta antes de aplicarle el algoritmo que la máquina realiza).

3. Sean A alfabeto y $Q = \{q_0, q_1, \ldots, q_{n-1}, q_f\}$.
   Sea δ función en la que, para todos los $i \in \{0, 1, \ldots, n-2\}$ y $a \in A$, $\delta(a, q_i) = (a, q_{i+1}, \rightarrow)$, $\delta(a, q_{n-1}) = (a, q_f, \rightarrow)$.
   Lo único que hace esta máquina es moverse n celdas a la derecha de donde comenzó.

4. Sea A alfabeto y $(a_1, a_2, \ldots, a_n)$ sucesión finita de símbolos de A. Sea $Q = \{q_0, q_1, \ldots, q_n, q_f\}$.
   Sea δ función en la que $\delta(q_0, 1) = (q_0, 1, \rightarrow)$, $\delta(q_0, 0) = (q_1, 0, \rightarrow)$, $\delta(q_1, 1) = (q_0, 1, \rightarrow)$, y para toda $i \in \{1, 2, \ldots, n-1\}$,
   $\delta(q_i, 0) = (q_{i+1}, a_i, \rightarrow)$ y $\delta(q_n, 0) = (q_f, a_n, \rightarrow)$.
   El lector de esta máquina se mueve a la derecha hasta encontrar dos 0's consecutivos y sobre el segundo 0 empieza a escribir, hacia la derecha, la sucesión de los a_i's. La escribirá completa si es que en sus n pasos a la derecha encuentra solamente 0's.

   Para el siguiente ejemplo se introducirá una forma de representar a la función de transición como una gráfica dirigida donde a cada estado le corresponde un vértice y a cada símbolo le corresponden varias aristas. Si una arista de símbolo a_h va de q_i a q_j quiere decir que el lector, al leer el símbolo a_h en estado q_i, cambiará al estado q_j. Para representar que en ese caso el lector remplazará a_h por a_w y se moverá a la izquierda se escribe $(a_h|a_w, \leftarrow)$.

5. Para cada alfabeto $A = \{a_1, a_2, \ldots, a_n, \kappa, \kappa'\}$, se hará una máquina que, al ser alimentada con una cadena de símbolos de $A \setminus \{\kappa, \kappa'\}$ seguida por los símbolos $\kappa'\kappa$ a su derecha, copia la cadena que va desde el símbolo donde se encuentra el lector hasta el símbolo que precede a κ', a partir de la κ que sucede a κ', hacia la derecha. Si la máquina fue alimentada con la cadena de símbolos c, la máquina regresa $c\kappa' c\kappa$.



El conjunto de estados se construye añadiendo tres estados q_i, q_{κ_i} y $q_{i'}$ por cada $i \in \{1, 2, \ldots, n\}$ de manera que esos tres estados funcionen como memoria mientras la máquina va (q_i) y regresa $(q_{i'})$. En el diagrama w es cualquier elemento de A y u y v son cualesquiera dos elementos de $A \setminus \{\kappa, \kappa'\}$.

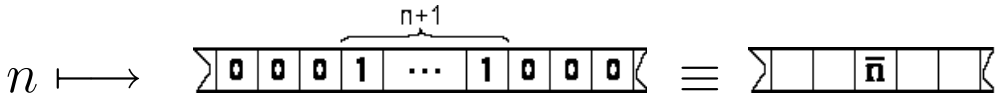
Es natural pensar en una *composición* de máquinas $N \circ M$, que consista en sustituir el último estado de M con el primero de N. Es decir, si M tiene m estados $\{q_0, q_1, \ldots, q_{m-2}, q_f\}$, y N tiene n estados $\{q_0, q_1, \ldots, q_{n-2}, q_f\}$, la

composición tiene $m+n-1$ estados y se construye así: a cada estado q_i de la máquina N cámbiesele el nombre a q_{m+i-1} y únanse los conjuntos de estados (borrando el estado final de M). También únanse las funciones de transición, con el cambio de nombre que se le hizo a N y sustituyendo las apariciones del estado final de M por q_{m-1}.

1.1.2. Como operadores de números

De particular interés para nosotros es el estudio de las *funciones numéricas* calculables por Máquinas de Turing pues, como espero se irá viendo, los números tienen un poder impresionante para representar otras cosas. Trabajaremos con el conjunto de los números naturales $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$ y estudiaremos las funciones de $\mathbb{N}^n$ en $\mathbb{N}$.

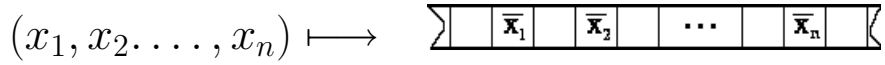
Primero que nada, se tiene que establecer algunas convenciones.
El natural n será representado como $n+1$ unos con únicamente ceros a la derecha y a la izquierda de los unos. Como no importa dónde empiecen los unos la representación de cada natural no es única.



Por simplicidad, cuando quiera representar $n+1$ unos en la cinta escribiré $\overline{n}$. Así mismo los ceros los omitiré de vez en cuando.

Extendiendo la representación de los naturales, las tuplas serán representadas como las coordenadas una tras otra con 0's como separadores.

Me referiré a las representaciones de n y de $(x_1, x_2, \ldots, x_n)$ en la cinta como $\overline{n}$ y $\overline{(x_1, x_2, \ldots, x_n)}$, respectivamente.



Se dice que una máquina computa una función parcial $f : I \to \mathbb{N}$, con $I \subset \mathbb{N}$, si cada vez que $f(x_1, \ldots, x_n) = y$, la máquina, al ser alimentada con $\overline{(x_1, \ldots, x_n)}$, computa $\overline{y}$, además de que, si $(x_1, \ldots, x_n) \notin I$, entonces f no realiza un cómputo o realiza sí realiza uno, pero en la cinta queda escrito algo que no representa a ningún número natural.

Ahora mostraré algunos ejemplos de máquinas que computan funciones numéricas, siempre suponiendo que el lector empieza sobre la celda que tiene el primer 1:

1.1 La suma $+ : \mathbb{N}^2 \to \mathbb{N}$:
Sean $Q = \{q_0, q_1, q_2, q_3, q_f\}$ y $A = \{0, 1\}$. Sea δ función de transición tal que $\delta(q_0, 1) = (q_0, 1, \rightarrow)$, $\delta(q_0, 0) = (q_1, 1, \rightarrow)$,
$\delta(q_1, 1) = (q_1, 1, \rightarrow)$, $\delta(q_1, 0) = (q_2, 0, \leftarrow)$, $\delta(q_2, 1) = (q_3, 0, \leftarrow)$ y
$\delta(q_3, 1) = (q_f, 0, \leftarrow)$.
Si la máquina recibe $\overline{(n, m)}$ como entrada, rastrea el 0 intermedio y escribe un 1 en su lugar. Luego quita dos 1's del final, por lo que quedan $(n + 1) + 1 + (m + 1) - 2 = n + m + 1$ unos, lo cual representa al número $n + m$.

1.2 La suma de un número con una constante c, $+_c : \mathbb{N} \to \mathbb{N}$:
Sea M_c la máquina que escribe c al final como una de las descritas en el ejemplo 4 de la sección anterior y sea M_+ la máquina que suma dos variables. La máquina $M_+ \circ R_1 \circ M_c$ escribe c al final, se regresa al primer 1 y luego aplica la suma del ejemplo anterior.
Esta construcción aclara que la máquina R_1 es importante pues, trabajando con funciones numéricas, rastrear el primer 1 con dos 0's a la izquierda significa rastrear el primer 1, de manera que dadas dos funciones $f : \mathbb{N}^n \to \mathbb{N}^m$ y $g : \mathbb{N}^m \to \mathbb{N}^k$ con máquinas M_f y M_g, la máquina que computa la composición $g \circ f$ es $M_g \circ R_1 \circ M_f$, siempre y cuando M_f termine el cómputo con el lector sobre el primer 1 o a su derecha.

El estudio de las máquinas de Turing que se está haciendo está enfocado en su capacidad de computar funciones numéricas pero, para probar el teorema de Rice, lo que es de interés es ver cómo es que algunas máquinas de Turing nos pueden dar información acerca de otras. La intención es mostrar que se puede representar a cada máquina de una manera que pueda ser codificada en la cinta. Así, surgen preguntas como ¿se podrá construir una máquina M que, alimentada con la representación de cualquier otra máquina N, M nos diga si N computa una función constante o no?
Este análisis comenzará en el capítulo 3.

1.2. Aritmética Formal

1.2.1. Lenguajes de predicados de primer orden

Un lenguaje de predicados de primer orden $\mathscr{L}$ consta de *símbolos* y reglas para concatenar los símbolos para obtener *fórmulas*. Dividiré a los símbolos en cinco clases basado en la función que estos cumplen:

1. Para representar a los objetos del universo de discurso se usan los símbolos de las siguientes tres clases:

   a) *constantes*: letras que representan objetos específicos.

   b) *variables*: letras que representan a los objetos de manera abstracta; es decir, una variable podría representar a cualquier objeto. Usaremos $x_1, x_2, x_3, \ldots$

   c) *funcionales*: dada cualquier cantidad de objetos, estas letras representan funciones aplicadas sobre los objetos dados. Cada funcional tiene una *aridad*, que es el número de objetos sobre el cual debe ser aplicado.

   Los *términos* serán todas aquellas expresiones que representen a los objetos. Estos se construyen recursivamente de la siguiente manera:

   - cada constante es un término
   - cada variable es un término
   - Una letra funcional f de aridad n aplicada a n términos $t_1, t_2, \ldots, t_n$ es un término. Dicho término se escribe $f(t_1, t_2, \ldots, t_n)$

2. Para representar a las relaciones entre los objetos se usan las letras *predicativas*. Igual que las letras funcionales, las predicativas tienen una aridad, y si $t_1, t_2 \ldots, t_n$ son términos y R es una letra predicativa de aridad n, $R(t_1, t_2 \ldots, t_n)$ representa a la relación entre los n objetos. A estas expresiones les llamamos *fórmulas atómicas*.

3. Para representar a los operadores lógicos se usan los símbolos de implicación ($\Rightarrow$), doble implicación ($\Leftrightarrow$), conjunción ($\wedge$), disyunción ($\vee$) y negación ($\neg$),

4. Para expresar la universalidad de una proposición ($\forall$) y la existencia de alguien que la cumpla ($\exists$). A estos símbolos se les llaman cuantificadores.

5. Como auxiliares se usan los paréntesis y las comas.

Las *fórmulas* de $\mathscr{L}$, se construyen recursivamente de la siguiente manera:

- Las fórmulas atómicas son fórmulas.
- Si α y β son fórmulas, $(\alpha \Rightarrow \beta)$, $(\alpha \Leftrightarrow \beta)$, $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$ y $(\neg\alpha)$ son fórmulas.
- Si α es una fórmula y x_i una variable, $(\forall x_i \alpha)$ y $(\exists x_i \alpha)$ son fórmulas.

Dentro de cualquier lenguaje deben estar las variables $x_1, x_2, x_3, \ldots$, los operadores lógicos, los cuantificadores y los paréntesis. De acuerdo a como se definan los conjuntos de constantes, letras funcionales y letras predicativas se definirá cada lenguaje particular. Para que el conjunto de fórmulas no sea vacío se pedirá que el conjunto de letras predicativas no lo sea.

Se dice que una variable x_i está *al alcance de un cuantificador* en α si existe una subfórmula $(\forall x_i \beta)$ o $(\exists x_i \beta)$ de α tal que x_i aparece en β.

Se dice que una fórmula α tiene a x_i como *variable libre* si x_i aparece en la fórmula y no está al alcance de ningún cuantificador. Cuando se escribe $\alpha(x_i)$ significa que α tiene a x_i como variable libre; también se dice que x_i aparece libre en α. Análogamente, $\alpha(x_{i_1}, x_{i_2}, \ldots, x_{i_n})$ es una fórmula en la que $x_{i_1}, x_{i_2}, \ldots, x_{i_n}$ aparecen libres.

Si $\alpha(x_{i_1}, x_{i_2}, \ldots, x_{i_n})$ es una fórmula y $t_1, t_2, \ldots, t_n$ son términos, entonces $\alpha(x_{i_1}|t_1, x_{i_2}|t_2, \ldots, x_{i_n}|t_n)$ es la fórmula que se obtiene de sustituir a x_{i_j} por t_j para todo j entre 1 y n.

Se le llama *enunciado* a una fórmula que no tiene variables libres. El concepto es relevante porque, intuitivamente, los enunciados son aquellas fórmulas sobre las que es razonable preguntarse sobre su valor de verdad. Por ejemplo, la pregunta '¿es verdadera la fórmula $x = 5$?' carece de sentido porque el objeto del que habla x es ambiguo. En cambio, si α es alguna de las fórmulas $4 = 5$, $\forall x(x = 5)$ ó $\exists x(x = 5)$, sí tiene sentido preguntarse sobre la verdad de α.

1.2.2. Teorías Formales

Una *teoría* T en un lenguaje $\mathscr{L}$ consta de un conjunto de fórmulas de $\mathscr{L}$ llamadas *axiomas*, y un conjunto de *reglas de deducción*, con las que se genera un conjunto de fórmulas de $\mathscr{L}$ llamadas *teoremas*. Si α es un teorema de T decimos que α *se deduce de* T y escribimos $T \vdash \alpha$.

Se definirá al conjunto de teoremas como el generado recursivamente por las siguientes reglas de deducción:

- Todo axioma es teorema.
- Regla *modus ponens*: Si α y $(\alpha \Rightarrow \beta)$ son teoremas entonces β también es teorema. Esta regla no hace más que describir la naturaleza de la implicación.
- Regla de *generalización*: Si α es un teorema con x_i como variable libre, $(\forall x_i \alpha)$ también es teorema. Esta regla es la que nos permite deducir que todos los objetos cumplen una propiedad, tomando un elemento arbitario y demostrando que éste la cumple.

Es por motivo de la regla de generalización que, a pesar de que las fórmulas con variables libres sufren de ambiguedad semántica, se conservan para el análisis de la verdad matemática que este método de deducción -a partir de axiomas- representa.

Las reglas son pocas porque, por lo general, el conjunto de axiomas contiene como *axiomas lógicos* a las siguientes clases de fórmulas:

1. Todas las *instancias de tautología*, que son aquellas fórmulas que se parten en subfórmulas que, vistas como proposiciones en un cálculo proposicional, forman una tautología.
2. $(\forall x_i \alpha \Leftrightarrow (\neg \exists x_i (\neg \alpha)))$ (caracterización de la existencia).
3. $((\forall x_i \alpha(x_i)) \Rightarrow \alpha(x_i|t))$ para cada x_i variable, α fórmula y t término que no tiene variables que quedan cuantificadas al meterlo en $\alpha(x_i)$. La fórmula $\alpha(x_i|t)$ es aquella que resulta de sustituir todas las apariciones libres de x_i por t (axioma de particularización).
4. $((\forall x_i(\alpha \Rightarrow \beta)) \Rightarrow (\alpha \Rightarrow (\forall x_i \beta)))$ para cada α y β fórmulas y x_i variable siempre y cuando α no contenga a x_i como variable libre.

Además, las teorías que tienen al símbolo de igualdad $(=)$ como letra predicativa tienen a los siguientes axiomas lógicos:

1. $\forall x_i (x_i = x_i)$ es un axioma.
2. Si x y y son variables y $\alpha(x, x)$ es una fórmula, $(\forall x(\forall y(x = y \Rightarrow (\alpha(x) \Rightarrow \alpha(x, x|y)))))$ es un axioma, donde $\alpha(x, x|y)$ es la fórmula que resulta de sustituir algunas, pero no necesariamente todas las apariciones de x por y.

Estas fórmulas nos permiten obtener muchas otras reglas lógicas conocidas. Para reforzar el caso daré algunos ejemplos de reglas derivadas de aplicar las reglas formales a los axiomas lógicos. Supóngase que T es una teoría con los axiomas lógicos dichos.

1. $T \vdash \alpha$ si y sólo si $T \vdash (\neg(\neg\alpha))$.
   Demostración. $(\neg(\neg\alpha))$ se deduce de α y la tautología $(\alpha \Rightarrow (\neg(\neg\alpha)))$ con modus ponens. Análogamente α se deduce de $(\neg(\neg\alpha))$ y la tautología $((\neg(\neg\alpha)) \Rightarrow \alpha)$.

2. $T \vdash (\alpha \wedge \beta)$ si y sólo si $T \vdash \alpha$ y $T \vdash \beta$.
   Demostración. α y β se deducen de $(\alpha \wedge \beta)$ y las tautologías $((\alpha \wedge \beta) \Rightarrow \alpha)$ y $((\alpha \wedge \beta) \Rightarrow \beta)$ con modus ponens, $(\alpha \wedge \beta)$ se deduce de α, β y la tautología $(\alpha \Rightarrow (\beta \Rightarrow (\alpha \wedge \beta)))$ aplicando modus ponens dos veces.

3. Si $T \vdash \alpha$ y β es una fórmula entonces $(\alpha \vee \beta)$ es un teorema.
   Demostración. Resulta de la tautología $(\alpha \Rightarrow (\alpha \vee \beta))$ y modus ponens.

4. $T \vdash (\alpha \Leftrightarrow \beta)$ si y sólo si $T \vdash (\alpha \Rightarrow \beta)$ y $T \vdash (\beta \Rightarrow \alpha)$.
   Demostración. Similar a las anteriores.

5. *Particularización:* Si $T \vdash (\forall x \alpha(x))$ entonces $T \vdash \alpha(x|t)$.
   Demostración. Se obtiene del axioma de particularización $((\forall x \alpha(x)) \Rightarrow \alpha(x|t))$ y modus ponens.

6. *Regla existencial:* Si t es un término, $T \vdash \alpha(t,t)$ y t no tiene a la variable x_i, entonces $T \vdash (\exists x_i \alpha(t|x_i, t))$.
   Demostración. Por axioma de particularización se tiene que la fórmula $((\forall x_i(\neg\alpha(t|x_i, t))) \Rightarrow (\neg\alpha(t,t)))$ es teorema y, por la tautología $((\beta \Rightarrow (\neg\gamma)) \Rightarrow (\gamma \Rightarrow (\neg\beta)))$ y modus ponens se tiene que $T \vdash (\alpha(t,t) \Rightarrow (\neg(\forall x_i(\neg\alpha(t|x_i, t)))))$. Si $T \vdash \alpha(t,t)$ entonces $T \vdash (\neg(\forall x_i(\neg\alpha(t|x_i, t))))$ y, por la caracterización de la existencia se tiene que $T \vdash (\exists x_i \alpha(t|x_i, t))$.

7. *Teorema de la deducción:* Si la fórmula α se deduce de la teoría T más una fórmula β, usando generalización sólo sobre fórmulas cuya deducción no depende de β o sobre fórmulas cuya deducción sí depende de β, pero la generalización es sobre una variable que no aparece libre en β, entonces $T \vdash \beta \Rightarrow \alpha$.
   Demostración. Por inducción sobre la cantidad de pasos en los que se usa a β o a una fórmula cuya deducción dependió de β para deducir a α: Si una fórmula γ se deduce sin usar a β, entonces por por modus ponens entre γ y la tautología $(\gamma \Rightarrow (\beta \Rightarrow \gamma))$ se tiene que $T \vdash (\beta \Rightarrow \gamma)$. Si γ

se deduce en $n+1$ pasos es porque se dedujo por modus ponens de un par de fórmulas η y $(\eta \Rightarrow \gamma)$ o por generalización de η, fórmulas para la cuales sólo se necesitaban n pasos o menos, por lo cual podemos aplicar hipótesis de inducción sobre ellas. Aplicando HI se tiene que $T \vdash (\beta \Rightarrow \eta)$. Para el caso en que γ se dedujo por modus ponens, se tiene que $T \vdash (\beta \Rightarrow \eta) \wedge (\eta \Rightarrow \gamma)$, y por la tautología de la transitividad de la implicación: $(((A \Rightarrow B) \wedge (B \Rightarrow C)) \Rightarrow (A \Rightarrow C))$ y modus ponens se tiene que $T \vdash (\beta \Rightarrow \gamma)$. Para el caso en que γ se dedujo por generalización sobre x_i de η, considérese la HI: $T \vdash (\beta \Rightarrow \eta)$, generalizado $T \vdash (\forall x_i(\beta \Rightarrow \eta))$, pero como β no tiene libre a x_i, se puede aplicar el axioma lógico 4 $T \vdash ((\forall x_i(\beta \Rightarrow \eta)) \Rightarrow (\beta \Rightarrow (\forall x_i \eta)))$. Aplicando modus ponens se tiene que $(\beta \Rightarrow (\forall x_i \eta))$, pero γ es $(\forall x_i \eta)$, así que ya se demostró lo que se quería.

8. *Prueba por contradicción:* Si la fórmula $(\alpha \wedge (\neg\alpha))$ se deduce de la teoría T más una fórmula β con las condiciones del teorema de la deducción, entonces $T \vdash (\neg\beta)$.
   Demostración. Por las condiciones del teorema de la deducción, se tiene que $T \vdash (\beta \Rightarrow (\alpha \wedge (\neg\alpha)))$. Por la tautología $(B \Rightarrow (A \wedge \neg A)) \Rightarrow \neg B$ y modus ponens se tiene que $T \vdash (\neg\beta)$.

9. $((\forall x_i(\alpha \Leftrightarrow \beta)) \Rightarrow ((\forall x_i \alpha) \Leftrightarrow (\forall x_i \beta)))$.

   Demostración.

   a) $(\forall x_i(\alpha \Leftrightarrow \beta))$ (Hipótesis para aplicar teorema de la deducción)

   b) $(\forall x_i \alpha)$ (Segunda hipótesis para aplicar teorema de la deducción, nótese que ésta se puede usar para hacer una fórmula $(\forall x_i \alpha) \Rightarrow B$ cuya demostración no dependa de $(\forall x_i \alpha)$, pero $(\forall x_i \alpha) \Rightarrow B$ seguirá condicionada a la primera hipótesis)

   c) α (Particularización de la segunda hipótesis)

   d) $(\alpha \Leftrightarrow \beta)$ (Particularización de la primera hipótesis)

   e) β (Modus ponens entre las dos anteriores)

   f) $(\forall x_i \beta)$ (Generalización; se puede hacer porque x_i no está libre en las hipótesis)

   g) $((\forall x_i \alpha) \Rightarrow (\forall x_i \beta))$ (Teorema de la deducción sobre la segunda hipótesis y la conclusión. Nótese que el teorema de la deducción dice que esta fórmula se puede deducir sin que se dependa de $(\forall x_i \alpha)$, sin embargo se sigue dependiendo de $(\forall x_i(\alpha \Leftrightarrow \beta))$)

   h) $((\forall x_i \beta) \Rightarrow (\forall x_i \alpha))$ (Se deduce de manera similar a la anterior)

i) $((\forall x_i(\alpha \Leftrightarrow \beta)) \Rightarrow ((\forall x_i\alpha) \Rightarrow (\forall x_i\beta)))$ (Teorema de la deducción sobre la deducción de 7. Nótese que ésta ya no depende de una fórmula fuera de la teoría T)

j) $((\forall x_i(\alpha \Leftrightarrow \beta)) \Rightarrow ((\forall x_i\beta) \Rightarrow (\forall x_i\alpha)))$ (Teorema de la deducción sobre la deducción de 8. Nótese que ésta tampoco depende de una fórmula fuera de la teoría T)

k) $((\forall x_i(\alpha \Leftrightarrow \beta)) \Rightarrow ((\forall x_i\alpha) \Leftrightarrow (\forall x_i\beta)))$ (De las tautologías $((A \Rightarrow B) \wedge (A \Rightarrow C)) \Rightarrow (A \Rightarrow (B \wedge C))$ y $((A \Rightarrow B) \wedge (B \Rightarrow A)) \Leftrightarrow (A \Leftrightarrow B)$)

10. *Sustitución por equivalentes:* Si $T \vdash (\alpha \Leftrightarrow \beta)$ entonces $T \vdash (\gamma[\alpha] \Leftrightarrow \gamma[\beta])$, donde $\gamma[\alpha]$ es una fórmula que tiene a α como subfórmula y $\gamma[\beta]$ es una fórmula que resulta de sustituir algunas de las apariciones de α por β en la fórmula γ
Demostración. Por inducción sobre la cantidad de símbolos lógicos $\neg$, $\Rightarrow$, $\Leftrightarrow$, $\wedge$, $\vee$ y $\forall$. Supóngase que $T \vdash (\alpha \Leftrightarrow \beta)$. Si γ no tiene ninguno de estos símbolos es porque es una fórmula atómica cuya única subfórmula es ella misma. En caso de ser así, $T \vdash (\gamma[\alpha] \Leftrightarrow \gamma[\beta])$ porque, si α es γ, $T \vdash (\alpha \Leftrightarrow \beta)$ y, si α no aparece en γ, $T \vdash (\gamma \Leftrightarrow \gamma)$.
En cualquier caso, si $\gamma[\alpha]$ es α, la demostración es trivial, así que se supondrá que no.

Caso 1. $\gamma[\alpha]$ es de la forma $\neg\eta$. Si α es subfórmula de $\gamma[\alpha]$ y es distina a $\gamma[\alpha]$ entonces α es subfórmula de η.

- Por hipótesis de inducción, $T \vdash \eta[\alpha] \Leftrightarrow \eta[\beta]$
- Por tautología $(A \Leftrightarrow B) \Rightarrow ((\neg A) \Leftrightarrow (\neg B))$ y modus ponens: $T \vdash (\neg\eta[\alpha]) \Leftrightarrow (\neg\eta[\beta])$
- Como $\gamma[\alpha]$ es distinta de α, $T \vdash (\gamma[\alpha] \Leftrightarrow \gamma[\beta])$

Caso 2. $\gamma[\alpha]$ es $\eta \Rightarrow \theta$. Si α es distinta de $\gamma[\alpha]$ entonces α está en η o en θ y $\gamma[\alpha]$ es, abusando un poco de la notación, $\eta[\alpha] \Rightarrow \theta[\alpha]$.

- Por Hipótesis de inducción $T \vdash (\eta[\alpha] \Leftrightarrow \eta[\beta])$ y
- $T \vdash (\theta[\alpha] \Leftrightarrow \theta[\beta])$
- Tómese $\eta[\alpha] \Rightarrow \theta[\alpha]$ como hipótesis para aplicar teorema de la deducción.
- Por tautología $((A \Rightarrow B) \wedge (B \Rightarrow C)) \Rightarrow (A \Rightarrow C)$ y modus ponens aplicado a ésta y a la conjunción de la hipótesis de inducción con la primera hipótesis para aplicar teorema de la

deducción, $\eta[\alpha] \Rightarrow \theta[\beta]$

- Lo mismo que en el anterior para concluir $\eta[\beta] \Rightarrow \theta[\beta]$
- Por teorema de la deducción $T \vdash ((\eta[\alpha] \Rightarrow \theta[\alpha]) \Rightarrow (\eta[\beta] \Rightarrow \theta[\beta]))$
- De forma similar se prueba la contraparte y se concluye $T \vdash ((\eta[\alpha] \Rightarrow \theta[\alpha]) \Leftrightarrow (\eta[\beta] \Rightarrow \theta[\beta]))$, lo cual es otra forma de escribir $T \vdash (\gamma[\alpha] \Leftrightarrow \gamma[\beta])$

Caso 3. $\gamma[\alpha]$ es de la forma $\eta \Leftrightarrow \theta$.
Su demostración es similar al caso 2.

Caso 4. $\gamma[\alpha]$ es de la forma $\eta \vee \theta$.
Para la demostración se hace la prueba para $((\neg\eta[\alpha]) \Rightarrow \theta[\alpha])$ usando los casos 1 y 2 y la tautología $(\neg A \Rightarrow B) \Leftrightarrow (A \vee B)$.

Caso 5. $\gamma[\alpha]$ es de la forma $\eta \wedge \theta$.
Para la demostración se hace la prueba para $(\neg((\neg\eta[\alpha]) \vee (\neg\theta[\alpha])))$ usando los casos 1 y 4 y la tautología $\neg(\neg A \vee \neg B)) \Leftrightarrow (A \wedge B)$.

Caso 6. $\gamma[\alpha]$ es de la forma $(\forall x_i \eta)$. Si α es distinta a $\gamma[\alpha]$ entonces $\gamma[\alpha]$ es $(\forall x_i \eta[\alpha])$.

- Por Hipótesis de Inducción $T \vdash (\eta[\alpha] \Leftrightarrow \eta[\beta])$
- Por generalización $T \vdash (\forall x_i(\eta[\alpha] \Leftrightarrow \eta[\beta]))$
- Por la propiedad anterior de $\forall$, $T \vdash ((\forall x_i \eta[\alpha]) \Leftrightarrow (\forall x_i \eta[\beta]))$, lo cual es otra forma de escribir
- $T \vdash (\gamma[\alpha] \Leftrightarrow \gamma[\beta])$

11. *Cambio de variables:* Si α y α' son fórmulas tales que α' resulta de sustituir en α a todas las apariciones cuantificadas de x_i por x_j, así como la que sucede al símbolo que la cuantifica ($\forall$, $\exists$), y x_j no aparece libre en α, entonces $T \vdash \alpha \Leftrightarrow \alpha'$.
Demostración. Sea γ la subfórmula de α en donde se sustituyen las variables. Entonces γ es de la forma $(\forall x_i \beta)$ o $(\exists x_i \beta)$. En el primer caso se tiene, por particularización, que β' (la que resulta de cambiar los x_i libres por x_j en β) y, por generalización, $(\forall x_j \beta')$. Aplicando teorema de la deducción para ambos lados se tiene que $T \vdash ((\forall x_i \beta) \Leftrightarrow (\forall x_j \beta'))$. Aplicando la regla de sustitución por equivalentes se tiene que $T \vdash \alpha \Leftrightarrow$

α'. En el caso en el que γ es $(\exists x_i \beta)$ se trabaja con $(\neg(\forall x_i(\neg\beta)))$ y se concluye lo mismo.

12. Algunas *propiedades de los cuantificadores* (sólo algunas se demostrarán como ejemplo):

   a) Si x_i no aparece libre en α, entonces $T \vdash (\alpha \Leftrightarrow (\forall x_i \alpha))$ y $T \vdash (\alpha \Leftrightarrow (\exists x_i \alpha))$.
   Demostración para $\forall$*:*
      1) $(\forall x_i(\alpha \Rightarrow \alpha))$ (Generalización de una tautología)
      2) $((\forall x_i(\alpha \Rightarrow \alpha)) \Rightarrow (\alpha \Rightarrow (\forall x_i \alpha)))$ (Axioma lógico 4)
      3) $(\alpha \Rightarrow (\forall x_i \alpha))$ (Modus ponens entre 2 y 3)
      4) $((\forall x_i \alpha) \Rightarrow \alpha)$ (Axioma de particularización)
      5) $(\alpha \Leftrightarrow (\forall x_i \alpha))$ (Regla derivada 4, aplicada a los dos anteriores)

   b) $T \vdash ((\forall x_i(\alpha \wedge \beta)) \Leftrightarrow ((\forall x_i \alpha) \wedge (\forall x_i \beta)))$
      1) $(\forall x_i(\alpha \wedge \beta))$ (Hipótesis para aplicar Teorema de la Deducción)
      2) $(\alpha \wedge \beta)$ (Particularización)
      3) α (De la tautología $(A \wedge B) \Rightarrow A$) y modus ponens.
      4) $(\forall x_i \alpha)$ (Generalización, posible porque α no tenía a x_i libre.)
      5) $(\forall x_i \beta)$ (Similar a la deducción de 4)
      6) $(\forall x_i \alpha) \wedge (\forall x_i \beta)$ (Conjunción de 4 y 5)
      7) $((\forall x_i(\alpha \wedge \beta)) \Rightarrow ((\forall x_i \alpha) \wedge (\forall x_i \beta)))$ (Teorema de la deducción)
      8) $(((\forall x_i \alpha) \wedge (\forall x_i \beta)) \Rightarrow (\forall x_i(\alpha \wedge \beta)))$ (Los mismos pasos de 6 a 1, usando la conclusión como hipótesis)
      9) $(((\forall x_i \alpha) \wedge (\forall x_i \beta)) \Leftrightarrow (\forall x_i(\alpha \wedge \beta)))$ (Propiedad de $\Leftrightarrow$ en 7 y 8)

   c) $T \vdash ((\exists x_i(\alpha \vee \beta)) \Leftrightarrow ((\exists x_i \alpha) \vee (\exists x_i \beta)))$

   d) $T \vdash ((\forall x_i(\alpha \Rightarrow \beta)) \Rightarrow ((\exists x_i \alpha) \Rightarrow (\exists x_i \beta)))$
      1) $(\forall x_i(\alpha \Rightarrow \beta))$ (Hipótesis)
      2) $(\alpha \Rightarrow \beta)$ (Particularización)
      3) $(\forall x_i(\neg\beta))$ (Hipótesis)
      4) $(\neg\beta)$ (Particularización)
      5) $((\neg\beta) \Rightarrow (\neg\alpha))$ (De la tautología $(A \Rightarrow B) \Leftrightarrow (\neg B \Rightarrow \neg A)$ y modus ponens con 1)

6) $(\neg\alpha)$ (Modus ponens entre 4 y 5)

7) $(\forall x_i(\neg\alpha))$ (Generalización, posible porque la hipótesis no tenía libre a x_i)

8) $(\forall x_i(\neg\beta)) \Rightarrow (\forall x_i(\neg\alpha))$ (Teorema de la deducción aplicado a la segunda hipótesis; nótese que ésta fórmula todavía es dependiente de la primera hipótesis, pero no de la segunda)

9) $((\neg(\forall x_i(\neg\alpha))) \Rightarrow (\neg(\forall x_i(\neg\beta))))$ (De la misma tautología)

10) $((\exists x_i\alpha) \Rightarrow (\exists x_i\beta))$ (Caracterización del existencial)

11) $((\forall x_i(\alpha \Rightarrow \beta)) \Rightarrow ((\exists x_i\alpha) \Rightarrow (\exists x_i\beta)))$ (Teorema de la deducción; nótese que ésta fórmula ya no depende de ninguna hipótesis)

e) Si x no aparece libre en α entonces $T \vdash ((\exists x(\alpha \Rightarrow \beta)) \Rightarrow (\alpha \Rightarrow (\exists x\beta)))$

Al lector que quiera profundizar más acerca del lenguaje, el cálculo de proposiciones y las teorías formales de primer orden, se le sugiere consultar [1].

1.2.3. Aritmética de Peano

A veces, cuando se habla de una teoría formal, no se sabe muy bien si se está definiendo un criterio para analizar el conjunto de verdades del universo de discurso, o si se está intentando construir, por medios totalmente formales (incluso medios con los cuales una computadora puede trabajar), un sistema lingüístico para expresar lo que ya se sabe acerca de los objetos. A veces no se entiende si el conjunto de objetos se define como *aquellas cosas que cumplen todo lo que la teoría dice* o si el conjunto está previamente definido y lo que se está analizando es la fuerza expresiva del lenguaje y la teoría formal.

Filosóficamente hay problemas al respecto. Sobre todo por los avances y las conclusiones parciales a las que, en el último par de siglos, se ha llegado, en el intento por formalizar las matemáticas. Por un lado, la teoría formal ZF ha mostrado un poder espectacular para deducir las *verdades* matemáticas a las que se llegó antes de la época de la formalización. Pero, por otro lado, Gödel, en 1931, demostró (de una manera formalizable en ZF) una limitación inevitable en las teorías formales. De hecho, Gödel inventó un método para construir, para cada teoría (manejable por una computadora y suficientemente expresiva), un enunciado intuitivamente verdadero, pero indemostrable por los métodos formales de la teoría (indemostrable por la com-

putadora). Esta limitación en las teorías formales con respecto a los métodos de deducción que usamos los humanos, podría parecer evidencia de que éstas se deben entender como *métodos para expresar las verdades que por cualquier otro modo se pueden deducir* y no como *métodos para deducir*. Esto no necesariamente quiere decir que, al nosotros poder concluir una verdad tal que la computadora no lo podía hacer, nosotros tengamos capacidades no formalizables (que no podría tener ninguna computadora), pues otro argumento válido es que la forma en la que nosotros deducimos la verdad de algunos de los enunciados de Gödel, a pesar de su indemostrabilidad, es derivada de otro método formal de deducción (o sea que podríamos programar a una computadora para reconocer y concluir la verdad de los enunciados de Gödel que nosotros reconocemos). Esto no contradice la prueba de Gödel pues, que para cierta clase de enunciados el método formal se pueda extender formalmente para demostrarlos, no quiere decir que el método se pueda extender para demostrar todos los enunciados de Gödel. En otras palabras, podría suceder (y seguramente sucede) que nosotros, al igual que las computadoras, estamos limitados para hacer un método que podamos aplicar para reconocer a todos los enunciados de Gödel, aunque algunos sí los reconozcamos. O sea, que tanto las teorías formales que se están estudiando, como cualquier otro método de deducción que nosotros nos inventemos, son igualmente limitados.

De todas maneras, al fijar una teoría formal, se corre el riesgo de que ésta no sea retórico-completa (y, en el caso de la aritmética, esto ocurre por el argumento de Gödel), en el sentido de que algún método de deducción igualmente convincente para nosotros, los humanos, no se pueda hacer dentro de la teoría. Es por esta razón que a la teoría se le tratará más como *método expresivo* que como *criterio de verdad* y que no se dudará en decir (y demostrar) proposiciones como 'si el natural n cumple la propiedad $P(n)$ entonces la aritmética formal demuestra que lo hace'; frases que, de entender la teoría como el *último criterio de verdad*, serían tautológicas. Lo que se hará es que se tratará a los objetos como si existieran fuera de la teoría, por más controvertido que esto sea, sólo por la posibilidad de que pudiéramos demostrar, con piedritas, u otra manera convincente, algo que no podríamos demostrar con la teoría formal.

La Aritmética de Peano (AP) es la teoría usual cuyo universo de discurso son los números naturales, es decir, los cuantificadores cuantifican sobre los números naturales. La fórmula $\forall x(\exists y(x < y))$ dice 'Para todo número natural existe otro más grande que éste '. El lenguaje de AP consta de los

siguientes símbolos. Escribiré al lado de cada símbolo su interpretación usual:

Letras predicativas:	Letras funcionales:	Constantes:
$=$ igualdad	s operación sucesor	0 cero
$<$ desigualdad	$+$ suma	
	$\times$ producto	

Aunque no hay un símbolo para representar a cada natural, la letra funcional s nos permite representar a cada n como el sucesor enésimo del 0. Por simplicidad, a dicha representación la escribiré como $\overline{n}$.

Las reglas de deducción son las definidas anteriormente. Además de los axiomas lógicos, es decir, los de la igualdad y las instancias de tautología, sus *axiomas propios* son los siguientes (con la interpretación usual):

1. s es inyectiva:
   $(s(x_1) = s(x_2) \Rightarrow x_1 = x_2)$
2. 0 es el único primer elemento:
   $((\neg s(x_1) = 0) \wedge ((\forall x_3(\neg x_2 = s(x_3))) \Rightarrow x_2 = 0))$
3. 0 es el neutro aditivo:
   $(+(x_1, 0) = x_1 \wedge +(0, x_1) = x_1)$
4. Definición recursiva de $+$:
   $(+(x_1, s(x_2)) = s(+(x_1, x_2)) \wedge s(+(x_1, x_2)) = +(s(x_1), x_2))$
5. Comportamiento del cero en $\times$:
   $(\times(x_1, 0) = 0 \wedge \times(0, x_1) = 0)$
6. Definición recursiva $\times$:
   $\times(x_1, s(x_2)) = +(\times(x_1, x_2), x_1)$
7. Definición de $<$:
   $(< (x_1, x_2) \Leftrightarrow (\exists x_3 + (x_1, s(x_3)) = x_2))$
8. Inducción:
   $((\forall x_1(\forall x_2(x_2 < x_1 \Rightarrow \alpha(x|x_2))) \Rightarrow \alpha(x|x_1)) \Rightarrow (\forall x_1 \alpha(x|x_1)))$
   Para cada fórmula $\alpha(x)$ donde x es una variable libre éste es un axioma.

A continuación se muestra una lista de algunos resultados conocidos de la

aritmética que se deducen formalmente en AP. En algunos se dará el ejemplo de la deducción pero se hará de una manera menos formal de lo que una teoría formal pide, con el motivo de que el lector no se pierda en la sintaxis. Sin embargo, se intentará que quede suficientemente claro que lo que se hace sí es formalizable. Entre otras cosas, se cambiará la notación de las operaciones a la notación usual, en ocasiones se evitará el uso excesivo de paréntesis que la teoría formal exige, se usarán variables distintas a las permitidas y las reglas derivadas se usarán sin preocupaciones, al igual que los teoremas que se demuestran en un ejemplo anterior.

Aunque varios de los resultados de la lista sí se usarán en capítulos posteriores, la ausencia de su demostración no debería preocupar demasiado al lector. Yo espero que los ejemplos que se dan sirvan para tranquilizar al lector nervioso, convenciéndolo de que la aritmética formal es una aproximación suficientemente decente a la aritmética de cuyos resultados fue convencido algún día el lector. Si aún con los ejemplos el descontento persiste, en la bibliografía hay material de consulta en donde aparecen las pruebas formales necesarias.[1]

- Inducción común: $\forall x(\alpha(0) \wedge (\alpha(x) \Rightarrow \alpha(s(x)))) \Rightarrow \forall x \alpha(x)$

  1. $\forall x[\alpha(0) \wedge (\alpha(x) \Rightarrow \alpha(s(x)))]$
     (Hipótesis para aplicar teorema de la deducción)
  2. $\forall y(y < x \Rightarrow \alpha(y))$
     (Hipótesis)
  3. $\forall k[x \neq s(k)] \Rightarrow x = 0$
     (Axioma de la unicidad del 0)
  4. $x = 0 \vee \exists k[x = s(k)]$
     (De la tautología $(A \vee B) \Leftrightarrow (\neg A \Rightarrow B)$ y la caracterización de la existencia)
  5. $x = 0 \Rightarrow \alpha(x)$
     (1 y propiedades de la igualdad)
  6. $x = s(k) \Rightarrow k + s(0) = x$
     (Definición de la suma)
  7. $k + s(0) = x \Rightarrow \exists i(k + s(i) = x)$
     (Regla existencial)
  8. $x = s(k) \Rightarrow k < x$
     (Tautología de la transitividad de la implicación)
  9. $k < x \Rightarrow \alpha(k)$
     (Particularización en la segunda Hipótesis)

10. $\alpha(k) \Rightarrow \alpha(s(k))$
(Particularización en la primera Hipótesis)

11. $x = s(k) \Rightarrow \alpha(s(k))$
(Transitividad de la implicación desde 7 hasta 9)

12. $\forall k[x = s(k) \Rightarrow \alpha(x)]$
(Propiedad de la igualdad, más generalización)

13. $\exists k(x = s(k)) \Rightarrow \exists k\alpha(x)$
(Propiedad del existencial)

14. $\exists k(x = s(k)) \Rightarrow \alpha(x)$
(Sustitución por equivalentes; nótese que α sí pudo haber tenido a k como variable libre, pero de igual manera se puede porque k se escogió arbitario, así que se puedo haber elegido de forma que no apareciera libre en α)

15. $[x = 0 \vee \exists k(x = s(k))] \Rightarrow \alpha(x)$
(De 3, 4 y 12, de la tautología $((A \Rightarrow C) \wedge (B \Rightarrow C)) \Rightarrow ((A \vee B) \Rightarrow C))$

16. $\alpha(x)$
(Modus ponens entre 3 y 13)

17. $\forall y(y < x \Rightarrow \alpha(y)) \Rightarrow \alpha(x)$
(Teorema de la deducción con la segunda hipótesis)

18. $\forall x \forall y(y < x \Rightarrow \alpha(y)) \Rightarrow \alpha(x)$
(Generalización)

19. $[\forall x \forall y(y < x \Rightarrow \alpha(y)) \Rightarrow \alpha(x)] \Rightarrow \forall x\alpha$
(Axioma de inducción)

20. $\forall x\alpha$ (Modus ponens en los anteriores)

21. $\forall x[\alpha(0) \wedge (\alpha(x) \Rightarrow \alpha(s(x)))] \Rightarrow \forall x\alpha(x)$
(Teorema de la deducción con la primera hipótesis)

- Conmutatividad de las operaciones.

1. $x_1 + 0 = x_1 = 0 + x_1$
(Por axioma 3 y propiedades de la igualdad)

2. $x_1 + x_2 = x_2 + x_1 \Rightarrow x_1 + s(x_2) = s(x_1 + x_2) = s(x_2 + x_1) = s(x_2) + x_1$
(Por propiedades de la igualdad y axioma 4)

3. $(x_1 + 0 = 0 + x_1 \wedge (x_1 + x_2 = x_2 + x_1 \Rightarrow x_1 + s(x_2) = s(x_2) + x_1)) \Rightarrow \forall x_2(x_1 + x_2 = x_2 + x_1)$
(Por inducción común)

4. $\forall x_2(x_1 + x_2 = x_2 + x_1)$
(De Modus Ponens entre 3 y la conjunción de 1 y 2)

5. $\forall x_1 \forall x_2 (x_1 + x_2 = x_2 + x_1)$
   (Por Generalización)

Para la multiplicación es similar.

- Asociatividad.
- Distributividad.
- Si n y m son naturales tales que $n = m$ entonces $AP \vdash \overline{n} = \overline{m}$.
  Demostración.
  Si $n = m$ entonces $\overline{n}$ y $\overline{m}$ tienen la misma cantidad de s's, o sea que son representados por el mismo término, entonces la aritmética, por el primer axioma de igualdad y particularización, prueba $\overline{n} = \overline{m}$.
- Si $n \neq m$ entonces $AP \vdash (\neg \overline{n} = \overline{m})$
  Demostración.
  Si $n \neq m$ quiere decir que, sin pérdida de generalidad, $n < m$, entonces $\overline{n}$ tiene menos s's que $\overline{m}$. En ese caso, podemos aplicar inyectividad de s n veces y obtener que $0 = \overline{m - n}$, pero como $m - n$ es un número positivo, $\overline{m - n}$ es $s(\overline{m - n - 1}$. Así, tendríamos que con AP más la fórmula $\overline{n} = \overline{m}$ se deduce $0 = s(\overline{m - n - 1})$ pero $AP \vdash (\forall x(\neg 0 = s(x)))$ y, por particularización, $AP \vdash (\neg 0 = s(\overline{m - n - 1}))$, así que, suponiendo $\overline{n} = \overline{m}$ se prueba $(\neg 0 = s(\overline{m - n - 1}) \wedge 0 = s(\overline{m - n - 1})$. Por la regla de la prueba por contradicción $AP \vdash (\neg \overline{n} = \overline{m})$.
- Si n, m y k son naturales y $n + m = k$, entonces $AP \vdash \overline{n} + \overline{m} = \overline{k}$.
  Demostración.
  Inducción sobre m: $AP \vdash \overline{n} + 0 = \overline{n}$ y $AP \vdash \overline{n} + \overline{m+1} = s(\overline{n + m})$. Pero $s(\overline{n + m})$ es $\overline{n + (m + 1)}$.
- Si n, m y k son naturales y $n \times m = k$, entonces $AP \vdash \overline{n} \times \overline{m} = \overline{k}$.
  Demostración.
  Inducción sobre m: $AP \vdash \overline{n} \times 0 = 0$ y $AP \vdash \overline{n} \times \overline{m+1} = \overline{n \times m} + \overline{n}$, pero $AP \vdash \overline{n \times m} + \overline{n} = \overline{n \times m + n}$. y $\overline{n \times m + n}$ es $\overline{n \times (m + 1)}$.
  De forma similar se prueba que, para todo n natural, $AP \vdash x_1 + \overline{n} = s(\cdots s(x) \cdots)$, con n s's. y que $x_1 \times \overline{n} = x_1 + x_1 + \cdots + x_1$, con n sumandos.
- Propiedades del orden:
  - Transitividad.
  - $x < y \Leftrightarrow s(x) < s(y)$
  - $x < y \Rightarrow (s(x) < y \vee s(x) = y)$
    1. $x < y$
       (Hipótesis para aplicar Teorema de la deducción)
    2. $s(x) < s(y)$
       (Modus Ponens entre la Hipótesis y la propiedad anterior del orden)
    3. $\exists k(s(x) + s(k) = s(y))$

(Equivalente a 2)

4. $s(x) + s(k) = s(y) \Rightarrow s(x) + k = y$
(Definición de la suma e inyectividad de s)

5. $\exists k(s(x) + s(k) = s(y)) \Rightarrow \exists k(s(x) + k = y)$
(Propiedades del existencial)

6. $(s(x) + k = y) \Rightarrow ((k = 0 \Rightarrow s(x) = y) \wedge (\neg(k = 0) \Rightarrow \exists z(s(x) + s(z) = y)))$
(El primero por propiedad del 0, el segundo por unicidad del 0 como primer elemento)

7. $(s(x) + k = y) \Rightarrow (s(x) = y \vee s(x) < y)$
(Por la tautología $((A \Rightarrow B) \wedge (\neg A \Rightarrow C)) \Rightarrow (B \vee C)$ aplicada a 6)

8. $\exists k(s(x) + k = y) \Rightarrow (s(x) = y \vee s(x) < y)$
(Propiedades del existencial)

9. $\exists k(s(x) + s(k) = s(y)) \Rightarrow (s(x) = y \vee s(x) < y)$
(De 5 y 8, por tautología $((A \Rightarrow B) \wedge (B \Rightarrow C)) \Rightarrow (A \Rightarrow C)$ y modus ponens)

- $\forall x(x = 0 \vee x > 0)$
- $x \neq 0 \Rightarrow 0 < x$
- $x < y \Rightarrow \neg(y < x \vee x = y)$

- Propiedades de divisibilidad. (Se escribe $x|y$ en lugar de $\exists k(x \times k = y)$)
  - $(x|y \wedge y|z) \Rightarrow x|z$
  - $x|y \Rightarrow x|y \times z$
  - $(x|y \wedge x|z) \Rightarrow x|y + z$
- Algoritmo de la división.
  1. $\forall x[(x \neq 0) \Rightarrow ((0 < x) \wedge (0 = x \times 0 + 0))]$
  $(((A \Rightarrow B) \wedge C) \Rightarrow (A \Rightarrow (B \wedge C)))$
  2. $\forall x[(x \neq 0) \Rightarrow \exists q \exists r((r < x) \wedge (0 = x \times q + r))]$
  (Regla existencial con propiedades de la existencia)
  3. $\forall x[(x \neq 0) \Rightarrow \exists q \exists r((r < x) \wedge (y = x \times q + r))]$
  (Hipótesis para aplicar Teorema de la Deducción)
  4. $\forall x[(x \neq 0) \Rightarrow \exists q \exists r((r < x) \wedge (s(y) = x \times q + s(r)))]$
  (Sustitución por equivalentes)
  5. $r < x \Rightarrow (s(r) < x \vee s(r) = x)$
  6. $\forall x[(x \neq 0) \Rightarrow \exists q \exists r((s(r) < x \vee s(r) = x) \wedge (s(y) = x \times q + s(r)))]$

(Sustitución por equivalentes)

7. $\forall x[(x \neq 0) \Rightarrow \exists q \exists r[((s(r) < x) \wedge (s(y) = x \times q + s(r))) \vee ((s(r) = x) \wedge (s(y) = x \times q + s(r)))]]$
(De la tautología $((A \vee B) \wedge C) \Leftrightarrow (A \wedge C) \vee (B \wedge C)$ y sustitución por equivalentes)

8. $\forall x[(x \neq 0) \Rightarrow \exists q \exists r[(s(r) < x) \wedge (s(y) = x \times q + s(r))] \vee \exists q \exists r[(s(r) = x) \wedge (s(y) = x \times q + s(r))]]$
(Propiedad del existencial y sustitución por equivalentes)

9. $\forall x[(x \neq 0) \Rightarrow \exists q \exists r[(s(r) < x) \wedge (s(y) = x \times q + s(r))] \vee \exists q \exists r[(s(r) = x) \wedge (s(y) = x \times q + x)]]$
(Propiedad de la igualdad)

10. $\forall x[(x \neq 0) \Rightarrow \exists q \exists r[(s(r) < x) \wedge (s(y) = x \times q + s(r))] \vee \exists q \exists r[(0 < x) \wedge (s(y) = x \times s(q) + 0)]]$
(Definición de la suma)

11. $\forall x[(x \neq 0) \Rightarrow \exists q \exists r[(r < x) \wedge (s(y) = x \times q + r)] \vee \exists q \exists r[(r < x) \wedge (s(y) = x \times s(q) + r)]]$
(Regla existencial)

12. $\forall x[(x \neq 0) \Rightarrow \exists q \exists r[(r < x) \wedge (s(y) = x \times q + r)]$
(De la tautología $A \Rightarrow (B \vee B) \Leftrightarrow (A \Rightarrow B)$ Y sustitución por equivalentes)

13. $\forall x[(x \neq 0) \Rightarrow \exists q \exists r((r < x) \wedge (y = x \times q + r))] \forall x[(x \neq 0) \Rightarrow \exists q \exists r[(r < x) \wedge (s(y) = x \times q + r)]$
(Teorema de la deducción del anterior y 3)

14. $\forall x[(x \neq 0) \Rightarrow ((0 < x) \wedge (0 = x \times 0 + 0))] \wedge (\forall x[(x \neq 0) \Rightarrow \exists q \exists r((r < x) \wedge (y = x \times q + r))] \Rightarrow \forall x[(x \neq 0) \Rightarrow \exists q \exists r[(r < x) \wedge (s(y) = x \times q + s(r))]]) \Rightarrow \forall y \forall x[(x \neq 0) \Rightarrow \exists q \exists r((r < x) \wedge (y = x \times q + r))]$
(Inducción común)

15. $\forall y \forall x[(x \neq 0) \Rightarrow \exists q \exists r((r < x) \wedge (y = x \times q + r))]$
(Modus ponens del anterior con la conjunción de 1 y 13)

16. La unicidad de q y r también se demuestra.

- Si se tiene una fórmula $\alpha(k)$ y se quiere que k represente a $|x - y|$, basta con sustituir cada fórmula atómica β en la que aparece k, por la expresión $((x + k = y \vee y + k = x) \Rightarrow \beta)$.

- Propiedades de anillo de las congruencias (donde $x \equiv y (\mathrm{mod}\ k)$ se traduce a que $k \mid |x - y|$), además de existencia de los debidos inversos.

- Si k es un natural, en AP se prueba la equivalencia entre $x < \overline{k+1}$ y $x = 0 \vee x = \overline{1} \vee \ldots \vee x = \overline{k}$.

- Toda propiedad no vacía tiene un mínimo que la cumple:
  $\exists x\alpha(x) \Rightarrow (\exists y(\alpha(y) \wedge (\neg\exists z(z < y \wedge \alpha(z)))))$

  1. $\forall w(w < x \Rightarrow [\alpha(w) \Rightarrow (\exists y(\alpha(y) \wedge (\neg\exists z(z < y \wedge \alpha(z)))))])$
     (Hipótesis 1)
  2. $\alpha(x)$
     (Hipótesis 2)
  3. $\neg(\exists z(\alpha(z) \wedge z < x)) \vee \exists z(\alpha(z) \wedge z < x)$
     (Tautología)
  4. $\alpha(x) \wedge \neg(\exists z(\alpha(z) \wedge z < x)) \Rightarrow \exists y(\alpha(x) \wedge \neg(\exists z(\alpha(z) \wedge z < x)))$
     (Regla existencial)
  5. $z < x \Rightarrow [\alpha(z) \Rightarrow (\exists y(\alpha(y) \wedge (\neg\exists z(z < y \wedge \alpha(z)))))]$ (Particularización en la hipótesis 1)
  6. $(z < x \wedge \alpha(z)) \Rightarrow (\exists y(\alpha(y) \wedge (\neg\exists z(z < y \wedge \alpha(z)))))$
     (Tautología)
  7. $\exists z(z < x \wedge \alpha(z)) \Rightarrow (\exists y(\alpha(y) \wedge (\neg\exists z(z < y \wedge \alpha(z)))))$
     (Propiedad del existencial)
  8. $\alpha(x) \Rightarrow (\exists y(\alpha(y) \wedge (\neg\exists z(z < y \wedge \alpha(z)))))$
     (De 3, 4, 7 y teorema de la deducción con la hipótesis 2. Llámesele $\beta(x)$ a esta fórmula)
  9. $\forall w(w < x \Rightarrow \beta(w)) \Rightarrow \beta(x)$
     (Teorema de la deducción con la hipótesis 1 y la conclusión anterior)
  10. $[\forall w(w < x \Rightarrow \beta(w)) \Rightarrow \beta(x)] \Rightarrow \forall x\beta(x)$
      (Axioma de inducción)
  11. $\forall x\beta(x)$
      (Modus ponens entre los dos anteriores)
  12. $\exists x\alpha(x) \Rightarrow (\exists y(\alpha(y) \wedge (\neg\exists z(z < y \wedge \alpha(z)))))$
      (Propiedad del existencial)

- Factorización única en primos.
- Teorema chino del residuo.

Como se puede ver, hay tres funciones que ya están explícitamente representadas (s, $+$ y $\times$), pero también se ve claramente que funciones como 'multiplicar por 3 y después sumar 2' se pueden representar $(y = +(\times(\overline{3}, x), \overline{2})$. Lo mismo pasa con las relaciones. Aunque $=$ y $<$ son las únicas representadas explícitamente, se pueden representar otras relaciones como 'ser menor o igual que el doble de' $((y < \times(\overline{2}, x)) \vee (y = \times(\overline{2}, x)))$.

Formalmente, si T es una teoría, se dice que una función $f : \mathbb{N}^n \to \mathbb{N}$ es *representable* en T si existe una fórmula $\alpha(x_1, \ldots, x_n, y)$ con $n+1$ variables libres tal que, si $f(k_1, k_2, \ldots, k_n) = m$, se cumplen las siguientes dos condiciones:

1. $T \vdash \alpha(x_i|\overline{k_i}, y|\overline{m})$.
2. $T \vdash (\forall y(\alpha(x_i|\overline{k_i}, y) \Rightarrow y = \overline{m}))$
   (unicidad).

En caso de que exista, se dice que la fórmula *representa* a la función.

Una relación $\mathscr{R} \subseteq \mathbb{N}^n$ es *representable* en T si existe una fórmula con n variables libres $\alpha(x_1, x_2, \ldots, x_n)$, tal que se cumplen las siguientes dos condiciones:

1. Si $(k_1, k_2, \ldots, k_n) \in \mathscr{R}$ entonces $T \vdash \alpha(x_1|\overline{k_1}, x_2|\overline{k_2}, \ldots, x_n|\overline{k_n})$
2. Si $(k_1, k_2, \ldots, k_n) \notin \mathscr{R}$ entonces $T \vdash (\neg\alpha(x_1|\overline{k_1}, x_2|\overline{k_2}, \ldots, x_n|\overline{k_n}))$.

En caso de que exista, se dice que la fórmula *representa* a la relación.

Dos conceptos importantes, a los que se hará referencia de nuevo en el capítulo 5, son el de consistencia y completez. Se dice que una teoría T es *consistente* si no sucede que exista una fórmula α tal que $T \vdash \alpha$ y $T \vdash (\neg\alpha)$ Se dice que T es *completa* si para todo enunciado α sucede que $T \vdash \alpha$ o bien que $T \vdash (\neg\alpha)$

Una observación relevante es que la interpretación que se le dé a los símbolos de un lenguaje no afecta para nada la estructura de una teoría con ese lenguaje. En otras palabras: que una fórmula sea o no teorema de una teoría no depende de lo que ésta signifique. Las fórmulas son sucesiones de símbolos y las reglas de deducción son simples operaciones simbólicas que nos dicen cómo generar unas sucesiones de símbolos a partir de otras. Esta observación es fundamental para la parte aritmética de esta tesis. Lo que se hará es salir un poco de la interpretación usual del lenguaje de la Aritmética de Peano. Se verá que, a pesar de que es un lenguaje construido para que los términos representen a los números naturales, también se pueden interpretar como símbolos, fórmulas y otros objetos usados en las teorías formales. Lo

que se verá es que, en este nuevo paradigma, operaciones simbólicas como las reglas de deducción y otras relaciones relevantes sí son representables en el lenguaje de la aritmética; es decir, el lenguaje de la Aritmética de Peano puede ser usado para estudiar la naturaleza de los mismos lenguajes y teorías.

Capítulo 2

Introducción a la Teoría de la Recursión

2.1. Funciones Recursivas

En esta tesis las funciones recursivas son importantes, por razones no ajenas, para probar el teorema de Rice y su versión para la aritmética. Para el teorema de Rice porque caracterizan a las funciones computables y para su versión aritmética porque hacen el vínculo entre ésta y las funciones computables, gracias a lo cual se muestra que ésta es capaz de funcionar como una teoría de la manipulación simbólica.

El conjunto de las funciones recursivas de $\mathbb{N}^n$ en $\mathbb{N}$ se construye recursivamente de la siguiente manera:

- La función *constante* 0, $C_0 : \mathbb{N} \to \mathbb{N}$ es recursiva.
- La función *sucesor* $s : \mathbb{N} \to \mathbb{N}$ es recursiva.
- Para cada n y k naturales, la *proyección* en la k-ésima coordenada $P_k^n : \mathbb{N}^n \to \mathbb{N}$ es recursiva.
- Para cada n natural y cada par de funciones recursivas $f : \mathbb{N}^n \to \mathbb{N}$ y $g : \mathbb{N}^{n+2} \to \mathbb{N}$, la función $h : \mathbb{N}^{n+1} \to \mathbb{N}$ que se *obtiene por recursión* de las ecuaciones $h(x_1, x_2, \ldots, x_n, 0) = f(x_1, x_2, \ldots, x_n)$ y $h(x_1, x_2, \ldots, x_n, s(x)) = g(x_1, x_2, \ldots, x_n, x, h(x_1, x_2, \ldots, x_n, x))$ es recursiva. Si $n = 0$, h se define en el 0 como una constante.
- Para cada n y k naturales, cada familia de funciones $g_1, g_2, \ldots, g_k : \mathbb{N}^n \to \mathbb{N}$ y cada función $f : \mathbb{N}^k \to \mathbb{N}$, la función $h : \mathbb{N}^n \to \mathbb{N}$ que

se *obtiene por sustitución de las g's en f* en que $h(x_1, x_2, \ldots, x_n)$ es igual a $f(g_1(x_1, x_2, \ldots, x_n), g_2(x_1, x_2, \ldots, x_n), \ldots, g_k(x_1, x_2, \ldots, x_n))$, es recursiva.

- Para cada número natural n y cada función recursiva $f : \mathbb{N}^{n+1} \to \mathbb{N}$, la función parcial que se *obtiene por minimalización* $\mu_y : \mathbb{N}^n \to \mathbb{N}$ tal que $\mu_y(x_1, x_2, \ldots, x_n) = min\{y \in \mathbb{N} : f(x_1, x_2, \ldots, x_n, y) = 0\}$, cuando dicho y exista; indefinida cuando no, es recursiva. La función $\mu_y(x_1, x_2, \ldots, x_n)$ también se escribe $\mu_y(f(x_1, x_2, \ldots, x_n, y))$.

Ejemplos:

1. La suma $+ : \mathbb{N}^2 \to \mathbb{N}$ se obtiene por recursión de las funciones recursivas $f : \mathbb{N} \to \mathbb{N}$ con $f = P_1^1$ (la identidad) y $g : \mathbb{N}^3 \to \mathbb{N}$, obtenida por sustitución de s en P_3^3: $g = s(P_3^3)$. Nótese que la sustitución en este caso es una composición simple. De esta manera $+(x_1, 0) = f(x_1) = P_1^1(x_1) = x_1$ y $+(x_1, s(x)) = g(x_1, x, +(x_1, x)) = s(P_3^3(x_1, x, +(x_1, x)))$ y este último es $s(+(x_1, x))$.
2. El producto $\times : \mathbb{N}^2 \to \mathbb{N}$ se obtiene por recursión de las funciones recursivas $f : \mathbb{N} \to \mathbb{N}$ con $f = C_0$ y $g : \mathbb{N}^3 \to \mathbb{N}$ donde $g = +(P_3^3, P_1^3)$, es decir, g se obtiene por sustitución de P_3^3 y P_1^3 en $+$. De esta manera $\times(x_1, 0) = f(x_1) = C_0(x_1) = 0$ y $\times(x_1, s(x)) = g(x_1, x, \times(x_1, x)) = +(P_3^3(x_1, x, \times(x_1, x)), P_1^3(x_1, x, \times(x_1, x))) = +(\times(x_1, x), x_1)$.
3. La exponenciación x^y se obtiene por recursión de $x^0 = 1$ y $x^{n+1} = x \cdot x^n$.
4. El factorial de n se obtiene por recursión de $0! = 1$ y $(n+1)! = n! \cdot (n+1)$.
5. El predecesor de n (cuando tiene) se obtiene por recursión de $pr(0) = 0$ y $pr(k+1) = k$.
6. La resta acotada por el 0: $n \dot{-} m$ se obtiene por recursión $n \dot{-} 0 = n$ y $n \dot{-} (k+1) = pr(n \dot{-} k)$.
7. La función sg tal que $sg(0) = 0$ y para todo k natural $sg(k+1) = 1$ se obtiene por sustitución donde $f = \dot{-}$ y $g_1 = P_1^1$ y $g_2 = pr$. O sea, $sg(k) = k - pr(k)$.
8. La función $\overline{sg}$ tal que $\overline{sg}(0) = 1$ y $\overline{sg}(k+1) = 0$ se obtiene también por sustitución: $\overline{sg}(k) = 1 \dot{-} sg(k)$.
9. El residuo de dividir n entre m: $rm(n, m)$ se obtiene por recursión de

$rm(0,m) = 0$ y $rm(k+1,m) = (rm(k,m)+1) \dot{-} m \cdot [(rm(k,m)+2) \dot{-} m)]$.

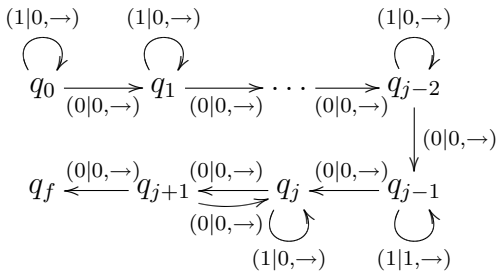
10. El cociente $n \div m$ se obtiene por recursión de $0 \div m = 0$ y $(k+1) \div m = k \div m + ([k+2 \dot{-} m \cdot (k \div m)] \dot{-} m)$.

A continuación se comenzarán a construir las demostraciones que ligan a las funciones recursivas con las máquinas de Turing y con la Aritmética de Peano. Primero se mostrará cómo se puede construir, para cualquier función recursiva, una máquina de Turing que la compute. Después se mostrará cómo cualquiera de éstas se puede representar dentro del lenguaje de AP.

2.1.1. Funciones Recursivas en MT

Primero se demostrará que cualquier función perteneciente a alguna de las tres clases de funciones recurisvas básicas (sucesor, constante 0 y proyecciones) es computable. Después se demostrará que cualquiera que se obtenga por recursión, sustitución o minimalización a partir de otras computables también es computable.

Lema 2.1.1. *Las funciones recursivas básicas son computables.*

- El ejemplo 1.2 de la sección 1.1 describe máquinas que suman constantes. Con $c = 1$ queda una MT que computa la función sucesor.
- Sea $Q = \{q_0, q_1, q_f\}$ conjunto de estados. Sea δ función tal que $\delta(1, q_0) = (0, q_0, \rightarrow)$, $\delta(0, q_0) = (0, q_1, \rightarrow)$, $\delta(1, q_1) = (0, q_0, \rightarrow)$ y $\delta(0, q_1) = (1, q_f, \rightarrow)$.
  Alimentada con $\overline{(n_1, n_2, \ldots, n_k)}$, la máquina borra todos los 1's y al final agrega uno. De esta forma, computa la constante 0.
- 

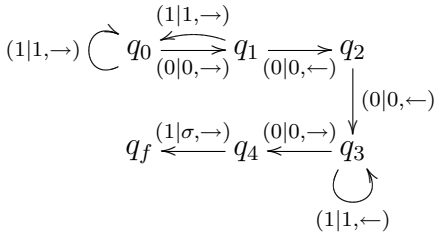
  Este diagrama describe a la MT que computa a P_j^k cuando es alimentada con $\overline{(n_1, n_2, \ldots, n_k)}$.

Lema 2.1.2. *Las funciones que se obtienen por recursión a partir de dos computables son computables.*

Demostración. Sean $g : \mathbb{N}^n \to \mathbb{N}$ y $f : \mathbb{N}^{n+2} \to \mathbb{N}$ computables con G y F las máquinas respectivas que las computan. Sea h la función que se obtiene por recursión de f y g.
Primero se describirá el funcionamiento de la máquina H que computa la función h.
Supongamos que se alimenta a la máquina con $\overline{(k_1, k_2, \ldots, k_{n+1})}$. La máquina deberá computar g con las primeras n coordenadas y después aplicar f debidamente k_{n+1} veces. Para esto, la máquina tendrá que hacer lo siguiente:

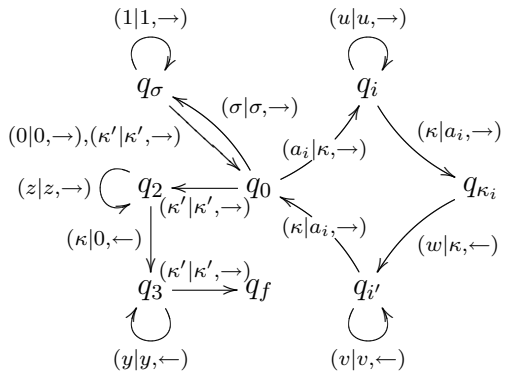
1. Colocar un símbolo σ sobre el primer 1 de la última coordenada. Esto sirve para marcar el paso de la recursión que se ha computado hasta el momento.
2. Copiar todo a la derecha de $\overline{(k_1, k_2, \ldots, k_{n+1})}$, excepto los unos que hay desde σ hasta el siguiente cero a su derecha.
3. Aplicar g sobre la segunda copia y asegurar que $\overline{g(k_1, k_2, \ldots, k_n)}$ quede a sólo un 0 de separación de la primera copia.
4. Recorrer el σ una celda hacia la derecha.
5. Copiar $\overline{(k_1, k_2, \ldots, k_{n+1}, k_{n+2})}$ al final, excepto los unos que hay entre σ y el primer cero a la derecha.
6. Aplicar f sobre la segunda copia y recorrer el cómputo a sólo un 0 de separación de $\overline{k_{n+1}}$ (borrando k_{n+2}).
7. Regresar al paso 4; si en dicho paso hay un 0 justo a la derecha de σ, romper el ciclo.
8. Borrar todo excepto la última coordenada escrita.

Ahora se esquematizará la construcción de la máquina que hace eso. Primero se describirán algunas MT necesarias.



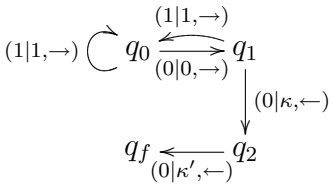
Este diagrama describe a la MT que computa el primer paso. Llámesele N a esta máquina.

Para el segundo paso se necesita una máquina copiadora como la descrita en el ejemplo 1.1.6 con alfabeto $A = \{0, 1, \kappa, \kappa', \sigma\}$ y un estado adicional q_σ, de manera que que reaccione al símbolo σ de forma que no lo copie ni a él ni a los unos que le siguen inmediatamente.



En el diagrama a_i es 0 o 1. w es cualquier símbolo, y u, v, y y z son 0 ó 1. El estado q_2 sirve para borrar el κ que la copiadora normal deja al final. El estado q_3 sirve para que el cómputo termine sobre el primer uno de la copia. A esta máquina llámesele C_σ.

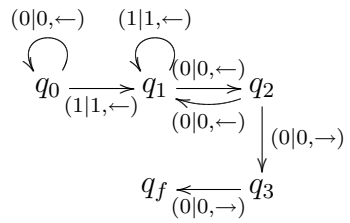
La siguiente máquina auxilia para el proceso de copiar:



Este diagrama describe a una MT que escribe $\kappa'\kappa$ al final de $\overline{(k_1, k_2, \ldots, k_{n+1})}$. Llámesele K a esta máquina.

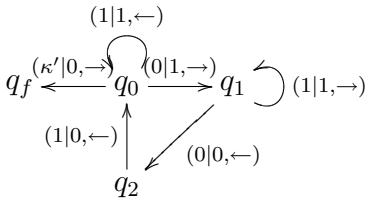
Recuérdese que la máquina copiadora deja al símbolo κ' entre la cadena original y la copia y termina justo a la derecha de éste, o sea, sobre el primer 1 de la segunda copia. De esta manera, se puede comenzar el cómputo de g justo ahí, siempre y cuando éste cómputo no utilice las celdas anteriores. Esto no causará problemas pues las máquinas que computan las funciones básicas no las usan y las que computan usando recursión, sustitución o minimalización, a partir de otras que no usan, se construirán de forma que tampoco las usen. Lo que no es cierto es que el resultado del cómputo comience en el mismo lugar en que comenzaba la entrada. Por esta razón se necesita una máquina que recorra el resultado hacia la izquierda hasta que éste quede como una coordenada más. Se supondrá que, cuando una máquina termina un cómputo, el lector queda sobre el primer 1 del resultado o a su derecha. De nuevo, esto se puede suponer porque las máquinas que computan las funciones básicas lo hacen, y las que se obtienen por recursión, sustitución o minimalización se

construirán de forma que lo hagan. Con dicha suposición se puede aplicar la siguiente máquina:



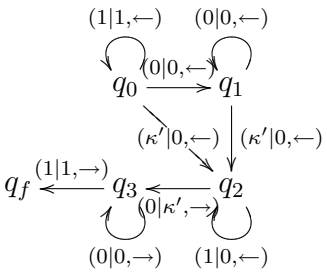
Este diagrama describe a la máquina cuyo lector se mueve a la izquierda hasta encontrar un 1 que tenga dos 0's consecutivos a la izquierda, y se detiene en dicho 1. Llámesele R_1 a esta máquina.

Gracias a esta máquina se puede empezar el siguiente cómputo con la suposición de que el lector está parado sobre el primer uno del cómputo de g.



Este diagrama describe a la máquina que, empezando sobre el primer 1 de una cadena de unos $\overline{k}$, que está a al menos una celda a la derecha de $\overline{(k_1, k_2, \ldots, k_{n+1})}$, recorre todo $\overline{k}$ hacia la izquierda hasta el κ' y pone un 0 en lugar de éste. Llámesele R a esta máquina.

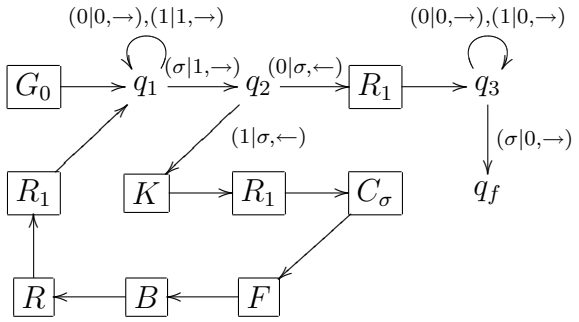
Sea $G_0 = R_1 \circ R \circ B \circ G \circ C_\sigma \circ R_1 \circ K \circ N$. Esta máquina, alimentada con $\overline{(k_1, k_2, \ldots, k_{n+1})}$, computa $\overline{(k_1, k_2, \ldots, k_{n+1}, g(k_1, k_2, \ldots, k_n))}$, con σ en lugar del primer 1 de la $(k+1)$-ésima coordenada. Hasta aquí van desarrollados los primeros 3 pasos. Los siguientes pasos son el ciclo de aplicar F e ir recorriendo el contador σ hasta que éste llegue al final de la última coordenada.
Primero hay que hacer la siguiente máquina auxiliar:



Este diagrama describe a la máquina que, empezando sobre una cadena de unos $\overline{k}$, que está a al menos una celda a la derecha de $\overline{(k_1, k_2, \ldots, k_{n+1}, k_{n+2})}\kappa'$, borra $\overline{k_{n+2}}\kappa'$ y pone κ' justo donde termina k_{n+1}. Llámesele B a esta máquina.

La máquina anterior hace que, después de computar h en cierto paso $k+1$ se pueda borrar el cómputo de h en el paso k.

La máquina H que computa h se describe en el siguiente diagrama:



El diagrama se interpreta de la manera natural. Las flechas de una máquina a otra son como la composición y las flechas entre máquinas y estados se deben intepretar como que salen del estado final o entran al estado inicial; todos los estados están 'ajenizados'.

□

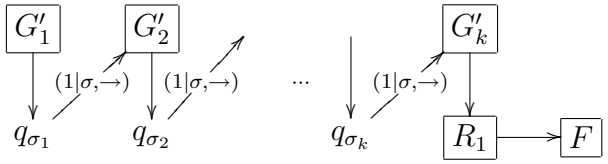
Lema 2.1.3. *Las funciones que se obtienen por sustitución a partir de otras computables son computables.*

Demostración. Sean $g_1, g_2, \ldots, g_k : \mathbb{N}^n \to \mathbb{N}$, $f : \mathbb{N}^k \to \mathbb{N}$ y $G_1, G_2, \ldots, G_k, F$ máquinas que las computan, respectivamente. Sea h la función que se obtiene por sustitución de ellas.

Sea $G_1' = R \circ G_1 \circ C_\sigma \circ R_1 \circ K$. Obsérvese que esta máquina, alimentada con $\overline{(k_1, k_2, \ldots, k_n)}$, lo copia, le aplica g_1 a la copia, y recorre el cómputo hasta la izquierda de manera que queda $\overline{(k_1, k_2, \ldots, k_n, g_1(k_1, k_2, \ldots, k_n))}$ escrito en la cinta. Si se pone σ en el primer 1 de $\overline{g_1(k_1, k_2, \ldots, k_n)}$ el proceso se puede repetir y la copiadora no tomará $\overline{g_1(k_1, k_2, \ldots, k_n)}$ en cuenta. Sea $G_i' = R \circ G_i \circ C_\sigma \circ R_1 \circ K$. Cada una de esas máquinas computará g_i sobre una segunda copia de $\overline{(k_1, k_2, \ldots, k_n)}$, ignorando los cómputos anteriores. Al final quedará

$$\overline{(g_1(k_1, k_2, \ldots, k_n), g_2(k_1, k_2, \ldots, k_n), \ldots, g_k(k_1, k_2, \ldots, k_n))}$$

y sólo basta aplicar f. El siguiente diagrama describe la máquina H que computa a la función que se obtiene por sustitución de los g_i y f.



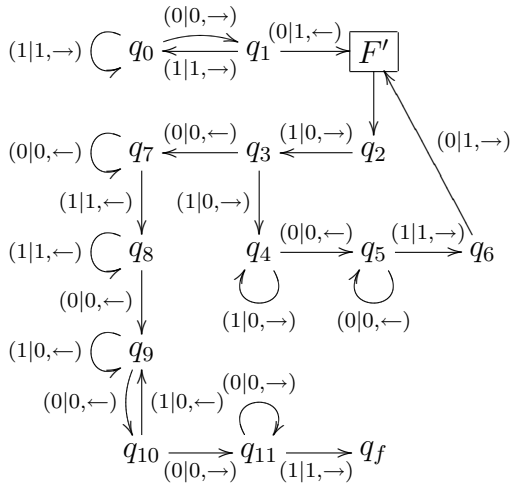
□

Lema 2.1.4. *Las funciones que se obtienen por minimalización a partir de otras computables son computables.*

Demostración. Sea $f : \mathbb{N}^{n+1} \to \mathbb{N}$ computable. Sea μ_y la que se obtiene por minimalización a partir de f.

Para computar h, la máquina necesita agregar un 0 como última coordenada, hacer una copia de lo que queda, y luego aplicar f sobre la segunda copia. Después, necesita verificar si el resultado es 0 y, en caso de que sí, borrar todo menos la última coordenada y terminar. En caso de que el resultado no sea 0 necesita agregar un 1 en la última coordenada y repetir el proceso.

Sea $F' = R \circ F \circ C \circ R_1 \circ K$. Con entrada $\overline{(k_1, k_2, \ldots, k_n, 0)}$, esta máquina la copia a la derecha, le aplica f y luego recorre el resultado hasta que éste queda como última coordenada. Termina sobre el primer 1 de $\overline{f(k_1, k_2, \ldots, k_n, 0)}$.



Los estados q_0 y q_1 colocan el 1 en la última coordenada. q_2 y q_3 deciden si el resultado de aplicar f fue 0. Si el resultado no fue 0, q_4, q_5 y q_6 borran el resto de $\overline{f(k_1, k_2, \ldots, k_n)}$, agregan un 1 a la última coordenada y regresan a F'. Si en el $(w+1)$-ésimo paso el resultado sí fue 0, los estados q_7, q_8, q_9, q_{10} y q_{11} borran todo salvo la última coordenada de $\overline{(k_1, k_2, \ldots, k_n, w)}$.

□

Proposición 2.1.1. *Todas las funciones recursivas son computables.*

Demostración. Las básicas son computables y las que se obtienen de otras computables por recursión, sustitución o minimalización también son computables. Por inducción todas son computables. □

2.1.2. Funciones Recursivas dentro de AP

Primero se demostrará que cualquiera de las tres funciones recursivas básicas (sucesor, proyecciones, constante 0) es representable en AP. Después se

demostrará que cualquiera que se obtenga por recursión, sustitución o minimalización a partir de otras representables también es representable. Nótese que la definición de representable sólo está dada para funciones totales y las funciones recursivas pueden ser parciales. Esto no importa porque el concepto de representabilidad no se usará más que para funciones totales.

Lema 2.1.5. *Las funciones recursivas básicas son representables:*

- $y = s(x_1)$ es representación de la función sucesor.
  Demostración. La primera condición de representabilidad se cumple porque, para cada natural m, $AP \vdash \overline{m+1} = s(\overline{m})$ pues los dos lados de la igualdad son $(m+1)$ aplicaciones de s al 0.
  La segunda condición se cumple porque $AP \vdash (y = s(\overline{m}) \Rightarrow y = \overline{m+1}))$ es una instancia de la tautología $A \Rightarrow A$.

- $y = 0$ es representación de la función constante 0.
  Demostración. La primera condición se cumple porque, para cada natural m, $AP \vdash (\overline{0} = 0)$ (El natural 0 se representa con el símbolo 0)
  La segunda condición se cumple porque $AP \vdash y = 0 \Rightarrow y = \overline{0}$.

- Para cada par de naturales $i \leq k$, la fórmula $x_{k+1} = x_i$ es representación de la proyección P_i^k.
  Demostración. La primera condición se cumple porque, para cada $(k+1)$-tupla de naturales $m_1, m_2, \ldots, m_k, m_{k+1}$ de forma que $m_{k+1} = m_i$, se tiene que $AP \vdash \overline{m_i} = \overline{m_i}$. La demostración de que la segunda condición se cumple es similar a las anteriores.

Para cualquier natural n, el lenguaje permite decir $\exists x_1 \exists x_2 \ldots \exists x_n$ pero nos enfrentamos con el problema de decir 'existe una sucesión con y elementos' donde y es una variable. Una expresión de este tipo es necesaria para representar a las funciones que se obtienen por recursión, pues el valor del $(n+1)$-ésimo término de una función que se obtiene por recursión depende de los n valores anteriores. Se quiere decir algo como 'para todo n existe una sucesión de n números tales que cada uno se obtiene de aplicar la función f al anterior'.

Para esto se hace uso de una manera de codificar sucesiones tan grandes como se quiera con sólo un par de números, de forma que sea equivalente decir 'existe un par de números...' y 'existe una sucesión con n elementos'. Para esto se aprovechará el teorema chino del residuo.

Primero obsérvese que, para $i < j \leq m$ naturales, los naturales $1 + (i + 1)m!$ y $1 + (j + 1)m!$ son primos relativos pues, si el primo p divide a ambos, entonces p divide a la diferencia $(j - i)m!$. Como p es primo, éste divide a $j - i$ o a $m!$. Si p divide a $m!$, como también divide a $1 + (i + 1)m!$, entonces divide al 1 y no es primo. Si p divide a $j - i$ entonces, como $j - i < m$, p divide a $m!$ y ya se vio que esto no es posible.

El teorema chino del residuo dice que dado un conjunto finito de primos relativos $p_1, p_2, \ldots, p_m$ y otro conjunto de naturales $k_1 < p_1, k_2 < p_2, \ldots, k_m < p_m$ existe $k < p_1 p_2 \ldots p_m$ tal que $x \equiv k(mod\ p_1 p_2 \ldots p_m)$ si y sólo si $x \equiv k_1(mod\ p_1)$, $x \equiv k_2(mod\ p_2)$, ..., $x \equiv k_m(mod\ p_m)$. Gracias a esto, basta con dar k y $m!$ para codificar una sucesión de m naturales $k_1 < 1 + (0 + 1)m!, k_2 < 1 + (1 + 1)m!, \ldots, k_m < 1 + (m - 1 + 1)m!$. Como $m!$ puede ser tan grande como se quiera, la sucesión puede ser tan larga como se quiera y los términos pueden ser tan grandes como se quiera. Si se quiere codificar la sucesión $k_1, k_2, \ldots, k_n$ tómese $m = max(k_1, k_2, \ldots, k_n, n)$.

Sea $\beta : \mathbb{N}^3 \to \mathbb{N}$ tal que $\beta(x, y, z) = rm(x, 1 + (z + 1)y)$. Nótese que con los naturales k, m, n descritos en el párrafo anterior y $i \leq n$ se tiene que $\beta(k, m!, i) = k_i$, así que, de tener una representación Bt de β se tendría que $Bt(\overline{k}, \overline{m!}, \overline{i}, x)$ dice 'x es el i-ésimo término de la sucesión $k_1, k_2, \ldots, k_n$'.
Sea $Bt(x_1, x_2, x_3, y)$ la fórmula que resulta de escribir en el lenguaje de AP:

$$(\exists x_4[x_1 = (1 + (x_3 + 1)x_2)x_4 + y \wedge y < (1 + (x_3 + 1)x_2)])$$

Bt sí es representación de β porque AP prueba algoritmo de la división. Además, $Bt(x_1, x_2, x_3, y)$ representa correctamente a la sucesión dada por x_1 y x_2, en el sentido de que, si $k_1, \ldots, k_n$ es la sucesión que el teorema chino del residuo da con m_1 y m_2, entonces $AP \vdash Bt(\overline{m_1}, \overline{m_2}, \overline{i}, \overline{k_i})$ y además de que a toda sucesión de naturales le corresponde una pareja m_1 y m_2. Esto es porque en AP se prueba el teorema chino del residuo. De este modo, decir que existen m_1 y m_2 es equivalente a decir que existe una sucesión.

Lema 2.1.6. *Dadas dos funciones representables, la obtenida por recursión a partir de ellas es representable.*

Demostración. Sean $g : \mathbb{N}^n \to \mathbb{N}$ y $f : \mathbb{N}^{n+2} \to \mathbb{N}$ representables y G y F sus representaciones respectivas. Sea $h : \mathbb{N}^{n+1} \to \mathbb{N}$ la función que se obtiene por recursión de g y f.

La fórmula $H(x_1, x_2, \ldots, x_{n+1}, x_{n+2})$ debe decir cuánto vale la función con $x_{n+1} = 0$. Como este valor depende de g se hace uso de su representación

G así: $G(x_1, x_2, \ldots, x_n, 0, x_{n+2})$. Para representar el comportamiento de la función con $x_{n+1} > 0$ se necesita decir que hay una sucesión de números tales que cada uno se obtiene del anterior a partir de la función f. Para esto se hace uso de Bt y de F de la siguiente manera:

$$\exists x \exists y \forall z_1 \forall z_2 \forall i (i < k \Rightarrow$$
$$[(Bt(x, y, i, z_1) \wedge Bt(x, y, s(i), z_2)) \Rightarrow F(x_1, x_2, \ldots, x_n, i, z_1, z_2)]) \wedge$$
$$[Bt(x, y, k, z_1) \Rightarrow F(x_1, x_2, \ldots, x_n, k, z_1, z)]$$

Si a esta fórmula le llamamos θ, la fórmula $H(x_1, x_2, \ldots, x_n, k, z)$ es:
$([k = 0 \Rightarrow G(x_1, x_2, \ldots, x_n, k, z)] \wedge (0 < k \Rightarrow \theta)$
A continuación se demuestra que H sí representa a h.

Supóngase que $h(k_1, \ldots, k_n, k) = m_k$. Si $k = 0$, es claro que en AP se prueba la fórmula $H(\overline{k_1}, \ldots, \overline{k_n}, \overline{k})$, porque G representa a g. Si $k > 0$, entonces hay una sucesión

$$m_0 = g(k_1, \ldots, k_n)$$
$$m_1 = f(k_1, \ldots, k_n, 0, m_0)$$
$$\vdots$$
$$m_k = f(k_1, \ldots, k_n, k-1, m_{k-1}).$$

Como F representa a f, ocurre que $AP \vdash F(\overline{k_1}, \ldots, \overline{k_n}, \overline{i}, \overline{m_i}, \overline{m_{i+1}})$ para cada $i < k$. Pero además, como ocurre que existen w_1 y w_2 tales que $Bt(w_1, w_2, \cdot, \cdot)$ representa correctamente a la sucesión de los m_j's, se tiene que

$$AP \vdash ((Bt(\overline{w_1}, \overline{w_2}, \overline{i}, z_1) \wedge Bt(\overline{w_1}, \overline{w_2}, \overline{i+1}, z_2)) \Rightarrow F(\overline{k_1}, \ldots, \overline{k_n}, z_1, z_2))$$

y también

$$AP \vdash Bt(\overline{w_1}, \overline{w_2}, k, z_1) \Rightarrow F(\overline{k_1}, \ldots, \overline{k_n}, z_1, \overline{m_k}))$$

Pero como ambas ocurren para cada $i < k$, se tiene que en AP se prueba la fórmula

$$\forall i (i < k \Rightarrow$$
$$((Bt(\overline{w_1}, \overline{w_2}, i, z_1) \wedge Bt(\overline{w_1}, \overline{w_2}, s(i), z_2)) \Rightarrow F(\overline{k_1}, \ldots, \overline{k_n}, z_1, z_2) \wedge$$
$$Bt(\overline{w_1}, \overline{w_2}, k, z_1) \Rightarrow F(\overline{k_1}, \ldots, \overline{k_n}, z_1, \overline{m_k}))$$

Para ver que $AP \vdash H(\overline{k_1}, \ldots, \overline{k_n}, \overline{k}, \overline{m_k})$, se generaliza sobre z_1 y z_2 y por la regla existencial, $\overline{w_1}$ y $\overline{w_2}$ se pueden quitar y agregar un existencial. Como se vio para los casos $k = 0$ y $k > 0$, sólo basta hacer la conjunción.

La unicidad se demuestra porque teorema chino del residuo dice que para cualesquiera x y y, $\beta(x, y, \cdot)$ determina una única sucesión. □

Lema 2.1.7. *Una función que se obtiene de otras representables por sustitución es representable.*

Demostración. Sean $g_1, g_2, \ldots, g_k : \mathbb{N}^n \to \mathbb{N}$, $f : \mathbb{N}^k \to \mathbb{N}$ y $G_1, G_2, \ldots, G_k, F$ sus representaciones respectivas. Sea $h : \mathbb{N}^n \to \mathbb{N}$ la función que se obtiene por sustitución de las otras. La fórmula $H(x_1, x_2, \ldots, x_n, y)$:

$$\forall y_1 \forall y_2 \ldots \forall y_k([G_1(x_1, x_2, \ldots, x_n, y_1) \wedge G_2(x_1, x_2, \ldots, x_n, y_2) \wedge \ldots \wedge G_k(x_1, x_2, \ldots, x_n, y_k)] \Rightarrow F(y_1, y_2, \ldots, y_k, y))$$

representa a h. A continuación se demostrará.

Supóngase que $h(r_1, \ldots, r_n) = m$ y considérese la sucesión

$$\begin{aligned} m_1 &= g_1(r_1, \ldots, r_n) \\ m_2 &= g_2(r_1, \ldots, r_n) \\ &\vdots \\ m_k &= g_k(r_1, \ldots, r_n) \end{aligned}$$

Para cada i, $AP \vdash (G_i(\overline{r_1}, \ldots, \overline{r_n}, y_i) \Rightarrow y_i = \overline{m_i})$, entonces

$$AP \vdash \forall y_1 \forall y_2 \ldots \forall y_k([G_1(\overline{r_1}, \ldots, \overline{r_n}, y_1) \wedge G_2(\overline{r_1}, \ldots, \overline{r_n}, y_2) \wedge \ldots \wedge G_k(\overline{r_1}, \ldots, \overline{r_n}, y_k)] \Rightarrow y_1 = \overline{m_1} \wedge y_2 = \overline{m_2}, \ldots, y_k = \overline{m_k})$$

pero $AP \vdash (y_1 = \overline{m_1} \wedge y_2 = \overline{m_2}, \ldots, y_k = \overline{m_k}) \Rightarrow F(y_1, y_2, \ldots, y_k, \overline{m})$, porque F representa a f. De esta manera se tiene que

$$AP \vdash \forall y_1 \forall y_2 \ldots \forall y_k([G_1(\overline{r_1}, \ldots, \overline{r_n}, y_1) \wedge G_2(\overline{r_1}, \ldots, \overline{r_n}, y_2) \wedge \ldots \wedge G_k(\overline{r_1}, \ldots, \overline{r_n}, y_k)] \Rightarrow F(y_1, y_2, \ldots, y_k, \overline{m})$$

La unicidad se demuestra para H porque, como se demuestra para g, se demuestra que la sucesión $r_1, \ldots, r_n$ es única y, como se demuestra para f, se demuestra que $f(r_1, \ldots, r_n)$ es único. □

Lema 2.1.8. *Una función total que se obtiene por minimalización a partir de una representable también es representable.*

Demostración. Sean $f : \mathbb{N}^{n+1} \to \mathbb{N}$ y F su representación. Sea $h : \mathbb{N}^n \to \mathbb{N}$ la obtenida por minimalización a partir de f. La fórmula $H(x_1, x_2, \ldots, x_n, y)$ que representa a h es

$$[F(x_1, x_2, \ldots, x_n, y, 0) \wedge (\neg \exists w[w < y \wedge F(x_1, x_2, \ldots, x_n, w, 0)])]$$

A continuación de demuestra esto.
Supóngase que $h(k_1, \ldots, k_n) = m$. Esto quiere decir que $f(k_1, \ldots, k_n, m) = 0$ y además, que m es el mínimo que anula a f. Como F representa a f, se tiene que

$$AP \vdash F(\overline{k_1}, \ldots, \overline{k_n}, \overline{m}, 0) \tag{2.1}$$

, pero además, para cada $w < m$, $AP \vdash (F(\overline{k_1}, \ldots, \overline{k_n}, \overline{w}, i) \Rightarrow (\neg i = 0))$, pero si ocurre para cada w, entonces sucede que

$$AP \vdash ((w < \overline{m} \wedge F(\overline{k_1}, \ldots, \overline{k_n}, w, i)) \Rightarrow (\neg i = 0))$$

entonces se tiene que

$$AP \vdash (i = 0 \Rightarrow [\neg(w < m \wedge F(k_1, \ldots, k_n, w, i))])$$

y de esta forma se obtiene que

$$AP \vdash (0 = 0 \Rightarrow [\neg(w < m \wedge F(k_1, \ldots, k_n, w, 0))])$$

y, por lo tanto,

$$AP \vdash \forall w[\neg(w < m \wedge F(k_1, \ldots, k_n, w, 0))]$$

de forma que, por la caracterización del existencial y en conjunción con 2.1, se puede concluir que

$$AP \vdash F(\overline{k_1}, \ldots, \overline{k_n}, \overline{m}, 0) \wedge (\neg\exists w(w < m \wedge F(k_1, \ldots, k_n, w, 0)))$$

La unicidad se demuestra porque en la aritmética se demuestra que cualesquiera dos mínimos son iguales. □

Proposición 2.1.2. *Toda función total recursiva es representable en AP.*

Demostración. Por los lemas anteriores e inducción sobre la cantidad de pasos que se necesitan para construir una función recursiva. □

De acuerdo a la definición, una función recursiva f se construye por una sucesión finita de funciones en que cada una es básica o se obtiene por recursión, sustitución, o minimalización de algunas de las anteriores, y la sucesión acaba en f. Claramente, una función no tiene una única sucesión de construcción, pero sería interesante tener un criterio algorítmico para comparar unas sucesiones de construcción con otras. Más adelante se expondrán algunos problemas relacionados con esta tarea.

Capítulo 3

Codificación de los Objetos

Los números naturales tienen un gran poder representacional. Cualquier sucesión finita de símbolos se puede codificar en un natural. Las codificaciones de este tipo son importantes para ambos lados del paralelismo de esta tesis. Para la aritmética, porque una fórmula es una sucesión de símbolos y acciones como la deducción de un teorema se pueden ver como una sucesión de fórmulas en la que cada una es axioma o se deduce de las fórmulas anteriores. Para las máquinas de Turing porque, para cada elemento de $A \times Q$, su imagen bajo la función de transición es un elemento de $A \times Q \times \{\leftarrow, \rightarrow\}$, de forma que la función de transición se puede codificar como una sucesión finita cuyos elementos sean todos los puntos de la gráfica de la función en $A \times Q \times A \times Q \times \{\leftarrow, \rightarrow\}$. Así mismo, cada cinta con una cantidad cofinita de celdas con 0's escritos se puede ver como una sucesión finita (que empieza en la primera celda que no es 0 y termina en la última celda que no es 0).

Codificar un conjunto X con números naturales significa pasar de X a $\mathbb{N}$ con una función inyectiva, de manera que dado un número que represente un objeto de X se pueda obtener de regreso a dicho objeto.
La teoría de funciones y relaciones recursivas se puede aplicar sobre la codificación de X y así se puede estudiar a éste. Esto es lo que se hará con los objetos de la aritmética formal y los objetos de las máquinas de Turing.
En ambos casos se aprovechará la factorización única en potencias de primos de los naturales para codificar a las sucesiones útiles.

La intención en codificar AP y MT con números naturales es mostrar que los objetos trabajados (números naturales), pueden representar a los mismos objetos con los que se les trabaja (términos, fórmulas, deducciones...; cintas,

máquinas, cómputos...). Así, la aritmética se puede *estudiar a sí misma*, y las máquinas dar información acerca de ellas mismas -las máquinas-.

3.1. Codificación de MT

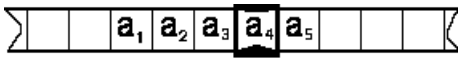
Como ya se mencionó, la función de transición de una máquina se puede ver como una sucesión de quíntuplas de $A \times Q \times A \times Q \times \{\leftarrow, \rightarrow\}$, así que para codificar, o aritmetizar, se necesita un número por cada elemento del alfabeto, uno por cada flecha y uno por cada estado. Por cada función recursiva se dio una forma para construir una máquina que la computa. Para esto solamente se usaron los símbolos 0, 1, σ, κ y κ', por lo que para codificar el alfabeto no se usarán más símbolos que estos.
A continuación se muestra la aritmetización $\sharp_m$ del lenguaje:

0	1	σ	κ	κ'	$\leftarrow$	$\rightarrow$	q_f
3	5	7	9	11	13	15	17

Para $0 \leq i$ y q_i un estado, se define $\sharp_m(q_i) = 19 + 2i$.

La función $\sharp_m$ se extiende a las quíntuplas y a las sucesiones de quíntuplas (que representan funciones de transición) de la siguiente manera: si $\varsigma_1\varsigma_2 \ldots \varsigma_n$ es una quíntupla de $A \times Q \times A \times Q \times \{\leftarrow, \rightarrow\}$ o una sucesión de dichas quíntuplas, $\sharp_m(\varsigma_1\varsigma_2 \ldots \varsigma_n) = 2^{\sharp_m(\varsigma_1)}3^{\sharp_m(\varsigma_2)} \ldots p_n^{\sharp_m(\varsigma_n)}$.

Además de estos objetos, $\sharp_m$ debe definirse sobre algo que diga qué hay escrito en la cinta, en qué celda se está parado el lector y en qué estado se encuentra éste. Esto se llama *descripción de la cinta*, y se representa como una sucesión de símbolos del alfabeto con un símbolo que representa al estado, sucediendo al símbolo del alfabeto sobre el cuál se encuentra el lector. Por ejemplo, si el lector está en estado q_i, la cinta



se representa con la sucesión $a_1a_2a_3a_4q_ia_5$. La función $\sharp_m$ se extiende a estas sucesiones igual que a las otras; el número correspondiente a la descripción de la cinta del ejemplo anterior es

$$2^{\sharp_m(a_1)}3^{\sharp_m(a_2)}5^{\sharp_m(a_3)}7^{\sharp_m(a_4)}11^{\sharp_m(q_i)}13^{\sharp_m(a_5)}$$

Nótese que, como los números de los símbolos son impares, los de las funciones de transición y descripiciones de cinta serán pares, pero con exponente impar

en el 2, de modo que se garantiza inyectividad, incluso si se considera en el dominio de $\sharp_m$ las sucesiones de sucesiones y todas las sucesiones construidas recursivamente con elementos otras sucesiones.

Dada esta codificación, las afirmaciones acerca de máquinas se puedan traducir en afirmaciones acerca de números. Este análisis se hará en la sección 3.3.

3.2. Codificación de AP

Los objetos para estudiar a una teoría formal son los símbolos, las sucesiones de símbolos, las sucesiones de fórmulas, etcétera. La aritmetización se define primero sobre los símbolos y se extiende recursivamente a sucesiones y sucesiones de sucesiones.

Aunque el lenguaje de AP ya se definió antes, aquí se hace de nuevo, con una cantidad de símbolos reducida que, aunque es suficiente, hubiera sido inconveniente para expresar con claridad lo que se expresó. También se simplificará la manera en que se forman los términos y las fórmulas atómicas.

Los símbolos que se quitarán son $\vee$, $\wedge$, $\exists$ y $\Leftrightarrow$ pues, para α y β fórmulas y t_1 y t_2 términos, estos ya son expresados con $(\neg\alpha) \Rightarrow \beta$, $\neg(\alpha \Rightarrow (\neg\beta))$, $\neg(\forall\neg(\alpha))$ y $\neg((\alpha \Rightarrow \beta) \Rightarrow (\neg(\beta \Rightarrow \alpha)))$ respectivamente.
En la formación de términos y fórmulas atómicas se quitarán de plano todos los paréntesis y las comas. Es decir, en lugar de $s(t)$, el sucesor se formará sólo agregando la s: st. La suma de dos términos t_1 y t_2 se formará como $+t_1t_2$ y el producto $\times t_1t_2$. La igualdad y el orden se formarán $= t_1t_2$ y $< t_1t_2$, respectivamente.

La función de aritmetización $\sharp_a$ en los símbolos está dada por la siguiente tabla

(	)	$\neg$	$\Rightarrow$	$\forall$	0	s	$+$	$\times$	$=$	$<$
3	5	7	9	11	13	15	17	19	21	23

y, si $i \geq 1$, $\sharp_a(x_i) = 23 + 2i$.

Si $\varsigma_1\varsigma_2\ldots\varsigma_n$ es una sucesión de objetos (símbolos o sucesiones de simbolos), $\sharp_a(\varsigma_1\varsigma_2\ldots\varsigma_n) = 2^{\sharp_a(\varsigma_1)}3^{\sharp_a(\varsigma_2)}\cdots p_n^{\sharp_a(\varsigma_n)}$ donde cada p_i es el i-ésimo primo. Esto extiende $\sharp_a$ recurisvamente. El teorema de la factorización única en primos implica que a dos sucesiones de símbolos distintas les correspondan números distintos.

Nótese que, por la misma razón que en la codificación de las máquinas, se puede garantizar la inyectividad de $\sharp_a$.

Codificar al lenguaje en números permite que las afirmaciones acerca del lenguaje se puedan traducir en afirmaciones acerca de números.

3.3. Recursividad en las codificaciones

Para verificar que, efectivamente, AP y de MT se pueden estudiar bajo sus gödelizaciones, se probará que las funciones, relaciones y propiedades de los objetos gödelizados, necesarias para su estudio, son recursivas. Que una relación o una propiedad sea recursiva significa que su función característica es recursiva. En general, la función característica de una relación R se denotará C_R, y el operador de minimalización se utilizará escribiendo $\mu_y R$ en lugar de $\mu_y(1 - C_R)$. Es decir, si R es una relación de $n+1$ argumentos $x_1, x_2, \ldots, x_n, y$, entonces $\mu_y R(x_1, x_2, \ldots, x_n, y)$ es igual al mínimo y tal que la relación $R(x_1, x_2, \ldots, x_n, y)$ se cumple. Nótese que si R es recursiva, $\mu_y R$ también lo es.
Se verán algunas reglas generales para construir relaciones y funciones recursivas.

Lema 3.3.9. *La relación de igualdad y la relación de orden son recursivas.*

Demostración. $C_=(x_1, x_2) = \overline{sg}(sg(x_1 \dot{-} x_2) + sg(x_2 \dot{-} x_1))$ y $C_<(x_1, x_2) = sg(x_2 \dot{-} x_1)$. □

Lema 3.3.10. *Si $R(x_1, x_2, \ldots, x_k)$ es una relación recursiva y $f_1, f_2, \ldots, f_k : \mathbb{N}^n \to \mathbb{N}$ es una familia de funciones totales recursivas, entonces la relación $R'(x_1, \ldots, x_n)$ igual a $R(f_1(x_1, \ldots, x_n), \ldots, f_n(x_1, \ldots, x_n))$ también es recursiva.*

Demostración. Si C_R es la función característica de R, es claro que la característica $C_{R'}$ es la que se obtiene por sustitución de las f_i's en C_R. □

Lema 3.3.11. *Si $R_1(x_1, x_2, \ldots, x_n)$ y $R_2(x_1, x_2, \ldots, x_n)$ son relaciones recursivas, las siguientes relaciones también son recursivas:*

1. $\neg R_1(x_1, x_2, \ldots, x_n)$

2. $R_1(x_1, x_2, \ldots, x_n) \Rightarrow R_2(x_1, x_2, \ldots, x_n)$
3. $R_1(x_1, x_2, \ldots, x_n) \wedge R_2(x_1, x_2, \ldots, x_n)$
4. $R_1(x_1, x_2, \ldots, x_n) \vee R_2(x_1, x_2, \ldots, x_n)$
5. La relación de argumentos $x_1, x_2, \ldots, x_{n-1}, k$:
$$(\exists x_n < k) R_1(x_1, x_2, \ldots, x_n)$$
6. La relación de argumentos $x_1, x_2, \ldots, x_{n-1}, k$:
$$(\forall x_n < k) R_1(x_1, x_2, \ldots, x_n)$$

Demostración.

1. Si C_{R_1} es la función característica de R_1, entonces $\overline{sg} \circ C_{R_1}$ es la característica de su negación.
2. $C_{R_1} \times C_{R_2} + 1 - C_{R_2}$ es la característica.

   3. y 4. están dadas por 1. y 2.

5. $sg(k \dot{-} \mu_{x_n}[R_1(x_1, x_2, \ldots, x_n) \vee (x_n \geq k)])$
   Nótese que se puede asegurar que μ_{x_n} está definida en todo el dominio gracias a que es una cuantificación acotada. De hecho, el resultado no es cierto si no se acota.
6. La relación $(\forall x_n < k) R_1(x_1, x_2, \ldots, x_n)$ es equivalente a la relación $\neg(\exists x_n < k)[\neg R_1(x_1, x_2, \ldots, x_n)]$, la cual es recursiva.

□

Ahora se estudiarán relaciones y funciones que tienen relevancia en las codificaciones de AP y a MT. Los tres lemas anteriores serán muy usados. Gracias a ellos se puede construir, para mostrar recursividad de relaciones y propiedades, expresiones en un lenguaje parecido al de la aritmética formal, pero en el cual los términos pueden ser formados con variables y cualquier función total recursiva aplicada a otros términos, las fórmulas atómicas se pueden construir con cualquier relación recursiva aplicada a términos, y los cuantificadores tienen que cuantificar sobre variables acotadas. Por ejemplo, la expresión $(\exists k < x)[2^k = x \wedge rm(x, 3) = 1]$ muestra la recursividad de la propiedad 'x es una potencia de 2 congruente con 1 módulo 3'.

Proposición 3.3.3. *Las siguientes funciones son recursivas:*

1. La función p tal que $p(n)$ es el n-ésimo primo. Siempre se denotará p_n en lugar de $p(n)$.

2. Si n se factoriza en primos $2^{a_0}3^{a_1}5^{a_2}\dots p_k^{a_k}$, la función de argumentos n y j, $(n)_j = a_j$ es recursiva. Para el número entendido como una sucesión de objetos, esta función es la extracción del j-ésimo objeto.

3. La función $\tau(n) = max\{k : p_k \text{ divide a } n\}$. Esta función calcula el tamaño de la sucesión.

4. Si $n = 2^{a_0}3^{a_1}\dots p_k^{a_k}$ y $m = 2^{b_0}3^{b_1}\dots p_l^{b_l}$, la función

$$n * m = 2^{a_0}3^{a_1}\dots p_k^{a_k}p_{k+1}^{b_0}p_{k+2}^{b_1}\dots p_{k+l+1}^{b_l}$$

es recursiva. Para n y m entendidos como sucesiones finitas, esta función es la concatenación de las sucesiones.

Demostración.

1. $p_0 = 2$, $p_{n+1} = \mu_k(k > p_n \wedge (\forall x < k)[rm(k, x) \neq 0])$.

2. Sea $R_j(n, k)$ la relación $rm(n, p_j^k) = 0 \wedge rm(n, p_j^{k+1}) \neq 0$. Así tenemos que $(n)_j = \mu_k R_j(n, k)$.

3. La relación $P(n, k)$ definida por $rm(n, p_k) = 0 \wedge (\forall l < n)(k < l \Rightarrow rm(n, p_l) \neq 0)$ dice 'p_k es el primo más grande que divide a n'. Así, $\tau(n) = \mu_k P(n, k)$.

4. Sea f la función definida por recursión tal que $f(n, m, 0) = n \cdot p_{\tau(n)+1}^{(m)_0}$ y $f(n, m, k+1) = f(n, m, k) \cdot p_{\tau(n)+k+1}^{(m)_k}$. En este caso $n*m = f(n, m, \tau(m))$.

□

3.3.1. Recursividad en el lenguaje de MT

Proposición 3.3.4. *Las siguientes propiedades, relaciones y funciones son recursivas:*

1. $SM(x)$: x es el número de un símbolo del alfabeto.

2. $E(x)$: x es el número de un estado.

3. $Q(x)$: x es el número de una quíntupla en $A \times Q \times A \times Q \times \{\leftarrow, \rightarrow\}$.

4. $Maq(x)$: x es el número de una máquina de Turing.
5. $\sharp E(x)$: función que da la cantidad de estados que tiene la máquina x.
6. $x \circ y$: función, tal que si x y y son números de máquinas, $x \circ y$ es el número de la composición.
7. $DC(x)$: x es el número de una descripción de cinta.
8. $DCC(x, y)$: x es el número de una descripción de cinta y y es el número de la sucesión de símbolos que la cinta de x tiene escritos.
9. $DCK(x, y)$: x es el número de una descripción de cinta que representa al natural y.

Demostración.

1. $x < 12 \wedge rm(x, 2) = 1$ (Porque los símbolos son números impares menores que 12).
2. $x > 16 \wedge rm(x, 2) = 1$ (Los estados son números impares mayores que 16).
3. $\tau(x) = 5 \wedge SM((x)_0) \wedge E((x)_1) \wedge SM((x)_2) \wedge E((x)_3) \wedge F((x)_4)$.
4. Se quiere construir una expresión que diga 'x es el número de una sucesión tal que cada elemento es quíntupla, hay una quíntupla con segundo elemento q_0 y cada pareja a, q_k aparece como primeros dos elementos en a lo más una quíntupla':

$$(\forall k \leq \tau(x))Q((x)_k) \wedge [(\exists l \leq \tau(x))((x)_l)_1 = 19] \wedge$$
$$[(\exists j, i < x)[((x)_j)_0 = ((x)_i)_0 \wedge ((x)_j)_1 = ((x)_i)_1] \Rightarrow i = j]$$

5. Suponiendo que para cada estado que la máquina tenga, ésta también tiene a todos los estados con índice menor (lo cual no es necesariamente cierto para una máquina, pero para el uso que se le hará sí lo es), el número de estados sólo será el número del estado más grande. La relación $y = \sharp E(x)$ está dada por la expresión:

$$(\exists i, j \leq \tau(x))[E(((x)_i)_j) \wedge$$
$$(\forall k, l \leq \tau(x))[E(((x)_k)_l) \Rightarrow ((x)_k)_l \leq ((x)_i)_j] \wedge y = (((x)_i)_j \dot{-} 19)/2]$$

   De este modo, $E(x) = \mu_y(y = \sharp E(x))$

6. La relación $Rec(x_1, x_2, p)$ que dice 'x_2 es la máquina que resulta de sustituir cada q_i de x_1 por q_{i+p}'es recursiva. La siguiente expresión lo muestra:

$$[Maq(x_1) \wedge Maq(x_2)] \wedge [\tau(x_1) = \tau(x_2)] \wedge (\forall n, m \leq \tau(x_1))$$
$$[([m = 1 \vee m = 3] \Rightarrow [((x_2)_n)_m = ((x_1)_n)_m + 2p]) \wedge$$
$$([m = 0 \vee m = 2 \vee m = 4] \Rightarrow [((x_2)_n)_m = ((x_1)_n)_m])]$$

   La relación $-Qf(y, y')$ que dice 'y y y' son iguales salvo porque y' no tiene estado final' es recursiva. La siguiente expresión lo muestra:

$$(\forall k \leq \tau(y))[[((y)_k)_3 = 17 \Rightarrow ((y')_k)_3 = 19 + 2(\sharp E(y) + 1)] \wedge$$
$$[((y)_k)_3 \neq 17 \Rightarrow (y)_k = (y')_k]]$$

   La función $x \circ y$ está dada por $\mu_{x'} Rec(x, x', \sharp E(y) + 2) * \mu_{y'} - Qf(y, y')$.

7. La siguiente expresión dice que todo exponente de los primos de x es el número de un símbolo o de un estado, pero hay exactamente uno de estado:

$$(\forall k \leq \tau(x))(SM((x)_k) \vee E((x)_k)) \wedge$$
$$[(\exists i \leq \tau(x))(E((x)_i) \wedge (\forall j \leq \tau(x))[E((x)_j)) \Rightarrow i = j)]]$$

8. La siguiente expresión dice que la sucesión x coincide con la sucesión de y salvo porque y no tiene al símbolo de estado:

$$(\tau(x) = \tau(y) + 1) \wedge (\forall j \leq \tau(y))(\forall i \leq \tau(x))$$
$$[E((x)_i) \Rightarrow ([j < i \Rightarrow (y)_j = (x)_j] \wedge [i \leq j \Rightarrow (y)_j = (x)_{j+1}])]$$

9. $[(\forall i \leq x)(\exists k < x)[DCC(x, k) \wedge (k)_i = 5 \wedge \tau(k) = y + 1]$ (La sucesión de símbolos escrita en la cinta consta de $y + 1$ unos).

□

3.3.2. Recursividad en el lenguaje de AP

Proposición 3.3.5. *Las siguientes propiedades y relaciones son recursivas:*

1. $V(x)$: x es el número de una variable.
2. $T(x)$: x es el número de un término.
3. $A(x)$: x es el número de una fórmula atómica.
4. $F(x)$: x es el número de una fórmula.

5. $x = \overline{k}$.

6. $Cuant(y, i, x)$: y es el número de una variable que aparece cuantificada en la i-ésima posición en la fórmula con número x.

7. $Lib(y, i, x)$: y es el número de una variable que aparece libre en la i-ésima posición de la fórmula con número x.

Demostración.

1. $x > 23 \wedge rm(x, 2) = 1$ es recursiva.

2. La idea en esta demostración es que se construya la expresión 'Existe una sucesión finita en la que cada elemento es variable, 0, o se obtiene de uno anterior agregando s al principio, o concatenando dos de ellos agregando + ó ×, y el último elemento es x':
$$(\exists z < x^x)(\forall k \leq \tau(z))$$
$$[(V(((z)_k)_0) \wedge \tau((z)_k) = 0) \vee (z)_k = 2^{\sharp_a(0)} \vee$$
$$(\exists i, j < k)([(z)_k = 2^{\sharp_a(s)} * (z)_i] \vee [(z)_k = 2^{\sharp_a(+)} * (z)_i * (z)_j]$$
$$\vee [(z)_k = 2^{\sharp_a(\times)} * (z)_i * (z)_j])] \wedge x = (z)_{\tau(z)}$$

3. $(\exists y < x)(\exists z < x)[(T(y) \wedge T(z)) \wedge (x = 2^{\sharp_2(<)} * y * z \vee x = 2^{\sharp_2(=)} * y * z)])$.

4. Se quiere construir la expresión 'Existe sucesión finita en que cada elemento es fórmula atómica, o es la concatenación de dos de los anteriores con un conectivo lógico, o es una fórmula que se obtiene de agregar un cuantificador universal a una de las anteriores, y el último elemento es x':
$$(\exists z < x^x)[(\forall k < \tau(z) + 1)(A((z)_k) \vee [(\exists i, j < k)$$
$$((z)_k = ((z)_i * 2^{\sharp_a \Rightarrow} * (z)_j) \vee (z)_k = (2^{\sharp_a \neg} * (z)_i) \vee$$
$$(\exists h < z)(V(h) \wedge (z)_k = 2^{\sharp_a(} * 2^{\sharp_a \forall} * 2^h * z_i * 2^{\sharp_a)}))])] \wedge (z)_{\tau(z)} = x]$$

5. $\tau(x) = k \wedge (\forall k < \tau(x))[(x)_k = \sharp_a(s)] \wedge (x)_{\tau(x)} = \sharp_a(0)$.

6. Se quiere construir la expresión 'existe una fórmula de la forma $(\forall x_i \alpha)$, subfórmula de x, tal que la i-ésima posición de x cae dentro de α y x_i es la variable con número y y aparece en la i-ésima posición':
$$F(x) \wedge V(y) \wedge [(\exists z < x)(\exists a, b < x)(F(z) \wedge (x = a * z * b) \wedge$$
$$(\exists c < z)[z = 2^{\sharp_a(} * 2^{\sharp_a \forall} * 2^y * c * 2^{\sharp_a)}]) \wedge$$
$$(\tau(a) < i \wedge \tau(a * z) > i) \wedge (x)_i = y)]$$

7. $F(x) \wedge V(y) \wedge (x)_i = y \wedge \neg Cuant(y, i, x)$.

□

A continuación se mostrará que las funciones y relaciones importantes para describir al cómputo (de MT) y a la demostración (de AP) son recursivas. Esto desembocará en que, como ambos trabajan a las funciones recursivas (a través del cómputo y la representación, respectivamente), habrá referencias inevitables de unas máquinas a otras y de unas fórmulas en otras.

3.3.3. Recursividad del cómputo en MT

Lema 3.3.12. *Las siguientes relaciones son recursivas.*

1. $DCN_0(x, n)$: x es una descripción de una cinta que tiene escrita a la representación de la tupla $(k_0, k_1, \ldots, k_n)$, para algunos naturales k_i's, el estado del lector es q_0 y éste se encuentra sobre el primer 1.
2. $Q_1(x, y, z)$: x es una quíntupla, y una descripción numérica, z es la descripción de cinta que resulta de aplicar la quíntupla x exactamente una vez a la cinta y (Si el estado del lector no coincide con el primer estado que aparece en la quíntupla o lo que está escrito en la celda no coincide con el primer símbolo de la quíntupla, entonces $z = y$ para que se cumpla la relación).

Demostración.

1. Se quiere construir la expresión 'existe una sucesión de tamaño n tal que el primer elemento es una sucesión de 1's (con el estado q_0 en la posición debida), y cada elemento resulta de agregar al anterior un 0 seguido por otra sucesión de 1's':

$$(\exists z < x)(\exists w < x^x)$$
$$[\tau(w) = n \wedge (\forall i \leq \tau(z))[(z)_1 = 19 \wedge (i \neq 1 \Rightarrow (z)_i = 5)] \wedge (w)_0 = z \wedge$$
$$(\forall i < \tau(w))[(\exists y < x)[(\forall j \leq \tau(y))[(y)_j = 5] \wedge (w)_{i+1} = w_i * 2^3 * y]] \wedge$$
$$x = (w)_{\tau(w)}]$$

2. Se quiere construir la expresión 'Si los primeros dos elementos de la quíntupla coinciden con el estado y el símbolo de la celda sobre la cual está el lector en la cinta y, entonces z es parecido a y, pero con el

cambio en la cinta, el cambio en la posición del lector, y el cambio en el estado del lector que la quíntupla indica':

$$Q(x) \wedge DC(y) \wedge DC(z) \wedge (\forall j < \tau(y))[E((y)_{j+1}) \Rightarrow$$
$$\{(((x)_1 \neq (y)_{j+1} \vee (x)_0 \neq (y)_j) \Rightarrow y = z) \wedge$$
$$(((x)_1 = (y)_{j+1} \wedge (x)_0 = (y)_j) \Rightarrow (\forall y' < y)(\forall z' < z)$$
$$[(DCC(y, y') \wedge DCC(z, z')) \Rightarrow$$
$$[(z')_j = (x)_2 \wedge (\forall k < \tau(z))[k \neq j \Rightarrow (y')_k = (z')_k] \wedge$$
$$((x)_4 = 13 \Rightarrow (z)_j = (x)_3) \wedge ((x)_4 = 15 \Rightarrow (z)_{j+2} = (x)_3)]])\}]$$

□

Sea $Comp(x, y)$ la relación: y es el número de una sucesión de descripciones de cinta en que cada una, al aplicarle la máquina x exactamente un paso, resulta en la descripción de cinta que le sucede y en la última de éstas, el estado del lector es q_f.

Proposición 3.3.6. *$Comp(x, y)$ es una relación recursiva.*

Demostración. La siguiente expresión lo muestra:

$$Maq(x) \wedge (\forall k \leq \tau(y))[(\forall i \leq \tau(x))[Q_1((x)_i, (y)_k, (y)_{k+1})]]$$

□

Corolario 3.3.1. *Existe una función recursiva $U : \mathbb{N}^2 \to \mathbb{N}$ tal que, si x es el número de una máquina y y es el número de una descripción de cinta, entonces $U(x, y) = M_x(y)$, donde $M_x(y)$ es la descripción de cinta que queda cuando la máquina con número x realiza un cómputo con la entrada y el lector dados por la descripción y (en caso de que la máquina realice un cómputo, en caso contrario, $U(x, y)$ no está definida).*

Demostración.

$$U(x, z) = (\mu_y[Comp(x, y) \wedge (y)_0 = z])_{\tau(\mu_y[Comp(x,y) \wedge (y)_0 = z])}$$

□

Lema 3.3.13. *Para cada n, la función $Desc_n(x_1, x_2, \ldots, x_n)$ que manda a la n-tupla en la descripción de cinta que tiene su representación, con el lector en q_0 en la primera posición, es recursiva.*

Demostración. Primero se mostrará que la relación asociada a la función: $R_{Desc_n}(x_1, \ldots, x_n, y)$ es recursiva. Esto se ve mediante la expresión:

$$(\exists y_1 < y)(\exists y_2 < y) \cdots (\exists y_n < y)$$
$$(\forall i \leq \tau(y_1))(\tau(y_1) = x_1 + 2 \wedge (y_1)_1 = 19 \wedge (i \neq 1 \Rightarrow (x)_i = 5)) \wedge$$
$$(\forall i \leq \tau(y_2))(\tau(y_2) = x_2 + 1 \wedge (x)_i = 5) \wedge \cdots \wedge$$
$$(\forall i \leq \tau(y_n))(\tau(y_n) = x_n + 1 \wedge (x)_i = 5) \wedge y = y_1 * 2^3 * y_2 * 2^3 * \cdots * 2^3 * y_n)$$

La función $Desc_n(x_1, x_2, \ldots, x_n)$ simplemente es $\mu_y R_{Desc_n}(x_1, \ldots, x_n, y)$. □

Proposición 3.3.7. *Las clase de las funciones recursivas es idéntica con la clase de las funciones computables.*

Demostración. Basta probar que todas las funciones computables son recursivas.
Sea $f : \mathbb{N}^n \to \mathbb{N}$ computable y sea M máquina que la computa, con número k. La función $\mu_y[DCK(U(k, Desc_n(x_1, x_2, \ldots, x_n)), y)]$ es f y claramente es recursiva. □

3.3.4. Recursividad de la demostración en AP

Lema 3.3.14. *Las siguientes relaciones y propiedades son recursivas:*

1. $Ta(x)$: x es tautología.
2. $Axl(x)$: x es un axioma lógico.
3. $Ax_{AP}(x)$: x es uno de los axiomas propios de AP.
4. $Mp(x, y, z)$: z se deduce de x y y por Modus Ponens.
5. $Gen(x, z)$: z se deduce de x por generalización.

Demostración.

1. Se busca una expresar 'Existe una partición de α_x en subfórmulas de manera que cualquier asignación de valores de verdad de las subfórmulas resulta en una asignación del valor *verdadero* de la fórmula completa'.

$$(\exists w < x^x)(\forall u < x^x)$$
$$\{(w)_{\tau(w)} = x \wedge$$
$$(\forall k \leq \tau(w))[(u)_k \leq 1 \wedge (F((w)_k) \wedge$$

$$(\forall i, j < k)[((w)_k = (w)_i \Rightarrow (u)_k = (u)_i) \wedge$$
$$((w)_k = 2^{\sharp\neg} * (w)_i \Rightarrow (u)_k \neq (u)_i) \wedge$$
$$((w)_k = (w)_i * 2^{\sharp\Rightarrow} * (w)_j \Rightarrow (u)_k = 1 \dot{-} (u)_j \cdot (1 \dot{-} (u)_i))]]$$
$$\Rightarrow (u)_{\tau(w)} = 1\}$$

2. Para cada uno de los axiomas lógicos distintos a *ser tautología* y a la caracterización de la existencia, ésta es la forma de mostrar su recursividad:

   Ax. 3 Primero se verá que la relación $s(z, z', y, t)$, que se cumple si las dos fórmulas son tales que z' se obtiene de z sustituyendo exactamente una aparición libre de la variable con número y por el término t, es recursiva. Luego se verá la relación $S(z, z', y, t)$ similar, pero en la cual se sustituye todas las y's libres por t's es recursiva, haciendo uso de la herramienta 'existe una sucesión finita tal que cada elemento de ésta cumple la relación s con el anterior y esto se hace hasta que y ya no aparece libre'. La recursividad de la relación s se muestra así:

$$V(y) \wedge T(t) \wedge F(z) \wedge F(z') \wedge$$
$$(\exists a, b < z)[Lib(y, \tau(a) + 1, z) \wedge (z = a * 2^y * b) \wedge (z' = a * t * b)]$$

   Usando la herramienta, la recursividad de la relación S se muestra por la expresión:

$$(\exists w < z^z \cdot z'^{z'})$$
$$[(w)_0 = z \wedge (w)_{\tau(w)} = z' \wedge (\forall i < \tau(w))[s((w)_i, (w)_{i+1}, y, t)] \wedge$$
$$(\forall j \leq \tau(z)) \neg Lib(y, j, z')]$$

   Pero, además, el esquema del axioma exige que t no tenga una variable que queda cuantificada al meterla en α, así que se necesita insertar la siguiente condición θ:

$$\neg(\exists i \leq \tau(t))[((t)_i = y) \wedge (\exists j \leq \tau(w))(\exists f, g < w)(\exists a, b < w)$$
$$[F(f) \wedge F(g) \wedge (f = 2^{\sharp(} * 2^{\sharp\forall} 2^{\sharp y} * a * 2^y * b * 2^{\sharp)}) \wedge$$
$$(g = 2^{\sharp(} * 2^{\sharp\forall} 2^{\sharp y} * a * t * b * 2^{\sharp)})] \wedge$$
$$(\exists c, d < w)[((w)_j = c * f * d) \wedge ((w)_{j+1} = c * g * d)])$$

   Básicamente, la expresión dice que no existe un paso de la construcción en que una variable de t se cuantifique al meterla a la fórmula. La recursividad del axioma se muestra así:

$$(\exists z, z', y < x)(\exists w < z^z \cdot z'^{z'})(\exists t < z')$$
$$[(w)_0 = z \wedge (w)_{\tau(w)} = z' \wedge (\forall i < \tau(w))[s((w)_i, (w)_{i+1}, y, t)] \wedge \theta \wedge$$
$$(\forall j \leq \tau(z))[\neg Lib(y, j, z')] \wedge x = \sharp_a(\forall * 2^y * 2^{\sharp_a(} * z * 2^{\sharp_a\Rightarrow} * z' * \sharp_a[]))]]$$

Ax. 4 $$(\exists a, b < x)(\exists y < x)$$
$$[F(a) \wedge F(b) \wedge V(y) \wedge (\forall i \leq \tau(a))[\neg Lib(a, i, y)] \wedge$$
$$x = \sharp_a[((\forall] * 2^y * 2^{\sharp_a)} * a * 2^{\sharp_a \Rightarrow} * b * \sharp_a[)) \Rightarrow (] * a * \sharp_a[\Rightarrow (\forall] * 2^y * b * \sharp_a[)))]$$

Ax.$=_1$ $(\exists y < x)[V(y) \wedge x = \sharp_a[(\forall] * 2^y * 2^{\sharp_a =} * 2^y * 2^y * 2^{\sharp_a)}]$

Ax.$=_2$ $$(\exists y < x)(\exists y' < x)(\exists a < x)(\exists b < x)$$
$$[V(y) \wedge V(y') \wedge F(a) \wedge F(b) \wedge S(a, b, y, y') \wedge$$
$$x = \sharp_a[(\forall] * 2^y * \sharp_a[(\forall] * 2^{y'} * 2^{\sharp_a (=} * 2^y * 2^{y'} * 2^{\sharp_a \Rightarrow} * 2^{\sharp_a (} * a * 2^{\sharp_a \Rightarrow} * b * \sharp_a[))))]]$$

La propiedad *ser axioma lógico* es la disyunción de las cinco propiedades anteriores.

3. Algunos de los axiomas propios de la aritmética usan símbolos que no pertenecen al lenguaje sobre el que se está trabajando, así que lo que se aritmetizará es su traducción estándar. Los que son conjunciones se dividirán como si fueran dos axiomas, aprovechando que se tiene que es equivalente que dos fórmulas sean teoremas y que su conjunción sea teorema.

   Ax1. $x = \sharp_a[(= sx_1x_2 \Rightarrow = x_1x_2)]$

   Ax2. $x = \sharp_a[(\neg = sx_10)]$ y $x = \sharp_a[((\forall x_3(\neg = x_2sx_3)) \Rightarrow = x_20)]$

   Ax3. $x = \sharp_a[= +x_10x_1]$ y $x = \sharp_a[= +0x_1x_1]$

   Ax4. $x = \sharp_a[= +x_1sx_2s + x_1x_2]$ y $x = \sharp_a[= s + x_1x_2 + sx_1x_2]$

   Ax5. $x = \sharp_a[= \times x_100$ y $x = \sharp_a[= \times 0x_10]$

   Ax6. $x = \sharp_a[= \times x_1sx_2 + \times x_1x_2x_1]$

   Ax7. $x = \sharp_a[(< x_1x_2 \Rightarrow (\exists x_3 = +x_1sx_3x_2))]$ y
   $x = \sharp_a[(\exists x_3 = +x_1sx_3x_2) \Rightarrow < x_1x_2]$

   Ax8. $$(\exists y, y' < x)(\exists z, z' < x)$$
   $$[S(z, z', y, 2^{y'}) \wedge x = \sharp_a[((\forall] * 2^y * \sharp_a[(\forall] * 2^{y'} * \sharp_a[(<] * 2^{y'} * 2^y *$$
   $$2^{\sharp_a \Rightarrow} * z * \sharp_a[)) \Rightarrow] * z' * \sharp_a[) \Rightarrow (\forall] * 2^y * z * \sharp_a[))]][1]$$

4. $y = x * 2^{\sharp_a \Rightarrow} * z$

[1] La relación $S(x, x', v, t)$ es aquella que se cumple cuando x' se obtiene de sustituir todas las apariciones libres de la variable v por el término t en la fórmula x.

5. $(\exists y < x)[V(y) \wedge z = \sharp_a[(\forall] * 2^y * x * 2^{\sharp_a)}]$

□

Sea $Dem_{AP}(x)$ la propiedad de ser una demostración de AP, es decir, ser una sucesión de fórmulas en que cada una es axioma o se deduce de fórmulas anteriores por alguna de las reglas de deducción.

Proposición 3.3.8. *La propiedad $Dem_{AP}(x)$ es recursiva.*

Demostración. La siguiente expresión lo muestra:

$$(\forall k \leq \tau(x))[Axl((x)_k) \vee Ax_{AP}((x)_k) \vee$$
$$(\exists i, j < k)[Mp((x)_i, (x)_j, (x)_k) \vee Gen((x)_i, (x)_k)]]$$

□

Corolario 3.3.2. *Existe una función computable $Ded_{AP} : \mathbb{N} \to \mathbb{N}$ que enumera (con repeticiones) a todos los teoremas de AP.*

Demostración. Primero considérese la función definida por recursión D_w: $D_w(0) = \mu_y(Dem(y))$ y $D_w(n+1) = (\mu_y[Dem(y) \wedge y > D_w(n)])$. La función Ded_{AP} queda definida por $Ded_{AP}(n) = (D_w(n))_{\tau(D_w(n))}$. □

Capítulo 4

Teoremas Diagonales

Para este punto queda claro que se puede hacer el cambio de paradigma y se puede empezar a ver a AP más que como la teoría formal de los números naturales y a MT más que como una formalización de los algoritmos de manipulación simbólica. A ambas se les puede ver como si los objetos que trabajan (variables y cintas, respectivamente) representaran expresiones simbólicas, expresiones sobre las cuales se puede representar tanto la naturaleza de las teorías formales como la de las máquinas de Turing. Como se demostró en el capítulo anterior, a la misma manipulación simbólica, aquella que le respecta a la demostración y aquella que una máquina de Turing realiza, se puede representar en el lenguaje de AP o en el código de una máquina. De este modo una puede 'hablar' o 'estudiar' la demostración de fórmulas y la otra puede 'dar información' o 'trabajar' sobre el cómputo de máquinas.

4.1. Teorema de Recursión (MT)

Por simplicidad, se hará una enumeración de todas las máquinas de Turing, basada en la aritmetización, y se trabajará con dicha enumeración. Para cada n, se le llamará $\mathscr{M}_n$ a la máquina con número n y se escribirá $\mathscr{M}_n(x_1, \ldots, x_k)$ como el número que se obtiene de aplicar f a $(x_1, \ldots, x_k)$, en caso de que $\mathscr{M}_n$ compute f. Si dos máquinas de números p y q computan la misma función se escribirá $\mathscr{M}_p \equiv \mathscr{M}_q$.

La enumeración ϕ se obtiene por recursión de las ecuaciones $\phi(0) = \mu_y Maq(y)$ y $\phi(n+1) = \mu_y(Maq(y) \wedge \phi(n) < y)$. La inversa derecha está dada por $\phi^{-1}(x) = \mu_y(\phi(y) = x)$. De aquí en adelante, siempre que se mencione el número de una máquina, será en referencia a la numeración ϕ.

Sea $S_n^1(x,y)$ la función tal que, para todos los $x_1, x_2, \ldots, x_n$, se cumple que $\mathscr{M}_{S_n^1(x,y)}(x_1, x_2, \ldots, x_n) = \mathscr{M}_x(x_1, x_2, \ldots, x_n, y)$.

Lema 4.1.15. *La función S_n^1 es computable.*

Demostración. Se verá que S_n^1 es recursiva.
Si $\mathscr{M}_k$ es una máquina que, al ser alimentada con $\overline{(x_1, \ldots, x_n)}$ computa $\overline{(x_1, \ldots, x_n, y)}$, entonces $\mathscr{M}_{S_n^1(x,y)}$ debe ser igual a $\mathscr{M}_x \circ R_1 \circ \mathscr{M}_k$[1]. Basta probar que existe una función computable F tal que $F(y)$ es el número de una máquina que escribe y al final de la cinta. Es claro que $S_n^1(x,y) = x \circ F(y)$, así que, sabiendo que existe la F computable, se tendría que S_n^1 es computable. Nótese que la máquina dada por la sucesión de quíntuplas

$$1q_0 1q_0 \rightarrow\ 0q_0 0q_1 \rightarrow\ 1q_1 1q_0 \rightarrow$$
$$0q_1 1q_2 \rightarrow\ 0q_2 1q_3 \rightarrow \cdots 1q_{y+1} 1q_{y+2} \rightarrow$$

es la máquina buscada. A la parte que no depende de la y le podemos extraer su número $r = \sharp_m(1q_0 1q_0 \rightarrow\ 0q_0 0q_1 \rightarrow\ 1q_1 1q_0 \rightarrow)$ y definimos recursivamente $F(0) = r * (2^{\sharp_m(0q_1 1q_2 \rightarrow)})$ y $F(n+1) = F(n) * (2^{\sharp_m(0q_{n+2} 1q_{n+3} \rightarrow)})$. Así, F es recursiva y, por lo tanto, computable.

□

Teorema 1. *(Teorema de Recursión [Kleene, 1938])*
Para cada función computable $f : \mathbb{N}^{n+1} \rightarrow \mathbb{N}$, existe un natural e tal que, para todos los $x_1, x_2, \ldots, x_n$, se tiene que

$$f(x_1, x_2, \ldots, x_n, e) = \mathscr{M}_e(x_1, x_2, \ldots, x_n)$$

Demostración. Considérese a d el índice de una máquina que computa la función de aridad $n+1$: $f(x_1, \ldots, x_n, S_n^1(x_{n+1}, x_{n+1}))$, es decir, que para cualquier n-tupla de naturales $x_1, x_2, \ldots, x_{n+1}$, se tenga que

$$f(x_1, \ldots, x_n, S_n^1(x_{n+1}, x_{n+1})) = \mathscr{M}_d(x_1, \ldots, x_n, x_{n+1})$$

. Por la construcción de S_n^1 se tiene la siguiente ecuación:

$$\mathscr{M}_{S_n^1(d, x_{n+1})}(x_1, \ldots, x_n) = \mathscr{M}_d(x_1, \ldots, x_n, x_{n+1})$$

Fijando $x_{n+1} = d$ se obtiene que

$$\mathscr{M}_{S_n^1(d,d)}(x_1, \ldots, x_n) = \mathscr{M}_d(x_1, \ldots, x_n, d) = f(x_1, \ldots, x_n, S_n^1(d,d))$$

Así, $e = S_n^1(d,d)$ es el número buscado. □

[1] La máquina R_1 es aquella que mueve el lector hacia la izquierda hasta el primer 1.

Nótese que, si e es dicho punto, lo que computa la máquina $\mathscr{M}_e$ es $f(x_1, x_2, \ldots, x_n, e)$, o sea, el cómputo de $\mathscr{M}_e$ depende de su propio código. Se puede decir que puede depender 'de la manera que se quiera' de su propio código, pues f es cualquier función computable.

A simple vista podría parecer que la diagonal $D(x) = S^1_n(x, x)$ es una máquina que trabaja con su propio código, como la que se enuncia en el Teorema de Recursión, pero no es así. La máquina $D(x)$ trabaja con el número x, pero es una máquina más compleja que la máquina con número x. De hecho, parecería imposible que una máquina primero escribiera su propio código, pues el código para escribir un número debe ser al menos igual de complejo que el número. El problema se resuelve fácilmente haciendo a la máquina d que primero diagonaliza a su último argumento. De este modo, $D(d)$ será una máquina que primero escribe d y luego procede a diagonalizarlo, obteniendo el número $D(d)$ (su propio número).

4.2. Teorema del Punto Fijo (AP)

En esta sección simplemente se le llamará α_n a la fórmula con número n.

Sea $\alpha_x(x_{i_1}, x_{i_2}, \ldots, x_{i_n})$ fórmula con n variables libres y sea $S^1(x, k)$ la función tal que $\alpha_{S^1(x,k)}$ es la fórmula que resulta de sustituir a todas las apariciones de la variable x_{i_n} por el término $\overline{k}$ en α.

Lema 4.2.16. *La función S^1 es representable.*

Demostración. Basta con demostrar que es recursiva.
Primero se verá la recursividad de la relación $R_{S^1}(x, k, y)$: 'x y y son números de fórmulas similares, con la única diferencia de que α_y tiene $\overline{k}$ cada vez que α_x tiene a x_{i_n}'. Para esto, se verá la recursividad de la relación $MaxV(x, v)$: 'De las variables que aparecen libres en x, v es la que tiene índice mayor '. Esto se muestra con la siguiente expresión:

$$(\exists i \leq \tau(x))Lib(v, i, x) \wedge (\forall k \leq \tau(x))[Lib((x)_k, k, x) \Rightarrow (x)_k \leq v]$$

La siguiente expresión muestra que $R_{S^1}(x, k, y)$ es recursiva:

$$(\forall v < x)[MaxV(x, v) \Rightarrow S(x, y, v, \sharp_a(\overline{k}))]^1$$

[1]La relación $S(x, x', v, t)$ es aquella que se cumple cuando x' se obtiene de sustituir todas las apariciones libres de la variable v por el término t en la fórmula x.

De este modo, $S^1(x,k) = \mu_y(R_{S^1}(x,k,y))$. □

Teorema 2. *(Teorema del Punto Fijo)*
Para cada fórmula $\sigma(x_{i_1},\ldots,x_{i_{n+1}})$, con $n+1$ variables libres tales que $i_1 < i_2 < \cdots < i_{n+1}$, existe un natural e tal que

$$AP \vdash (\sigma(x_1,\ldots,x_n,\overline{e}) \Leftrightarrow \alpha_e(x_1,\ldots,x_n))$$

donde $\alpha_e(x_1,\ldots,x_n)$ es la fórmula con número e.

Demostración. Sea $D(x) = S^1(x,x)$ la función diagonal, que resulta de sustituir a la variable libre de número más grande en α_x por $\overline{x}$. Podría parecer que, siempre que x sea el número de una fórmula con una variable libre, $D(x)$ será una fórmula que dice de sí misma lo que x decía de la variable libre, pero esto no es cierto; en realidad, $D(x)$ es una fórmula que se refiere a la fórmula x, que sí es muy parecida a $D(x)$, pero no es $D(x)$. Sin embargo, el poder de la aritmética formal para representar a las funciones recursivas permite que una fórmula se refiera a la diagonal de otra por medio de la representación de D. De este modo, si x fuera una fórmula que se refiriera a la diagonal de su variable libre a través de la representación de D, la fórmula $D(x)$ se estaría refiriendo a ella misma indirectamente, pues ella misma es la diagonal de x, que ahora sustituye a lo que era una variable libre.
Sea $\mathscr{D}(x,y)$ la representación de D. Considérese la fórmula α_m:

$$(\forall x_{i_{n+1}}(\mathscr{D}(x_w, x_{i_{n+1}}) \Rightarrow \sigma(x_{i_1},\ldots,x_{i_{n+1}})))$$

donde $i_j < w$ para todo $1 \leq j \leq n+1$.
Ahora considérese la diagonal de α_m, es decir, la fórmula $\alpha_{D(m)}$:

$$(\forall x_{i_{n+1}}(\mathscr{D}(\overline{m}, x_{i_{n+1}}) \Rightarrow \sigma(x_{i_1},\ldots,x_{i_{n+1}})))$$

Intuitivamente, $\alpha_{D(m)}$ dice 'Toda fórmula diagonal de α_m cumple $\sigma(x_{i_{n+1}})$', pero como $\alpha_{D(m)}$ es la única diagonal de α_m, entonces se puede interpretar como que dice 'Yo cumplo $\sigma(x_{i_{n+1}})$', así que si ocurre que $AP \vdash \alpha_{D(m)}$ debería ocurrir que AP demostrara $\sigma(x_{i_1},\ldots,x_{i_{n+1}}|\overline{D(m)})$ y así se tendría demostrado que $\alpha_{D(m)} \Rightarrow \sigma(x_{i_1},\ldots,x_{i_{n+1}}|\overline{D(m)})$. Y, al revés, si $\sigma(x_{i_1},\ldots,x_{i_{n+1}}|\overline{D(m)})$ se cumpliera se tendría que la diagonal de α_m la cumple, y entonces se tendría que toda diagonal la cumple (porque sólo hay una), y así quedaría demostrado $\sigma(x_{i_1},\ldots,x_{i_{n+1}}|\overline{D(m)}) \Rightarrow \alpha_{D(m)}$. Formalmente, estos son los pasos en la deducción en AP:

1. $(\forall x_{i_{n+1}}(\mathscr{D}(\overline{m}, x_{i_{n+1}}) \Rightarrow \sigma(x_{i_1}, \ldots, x_{i_{n+1}})))$
   Hipótesis $\alpha_{D(m)}$
2. $(\mathscr{D}(\overline{m}, \overline{D(m)}) \Rightarrow \sigma(x_{i_1}, \ldots, x_{i_n}, \overline{D(m)}))$
   Particularización
3. $\mathscr{D}(\overline{m}, \overline{D(m)})$
   $\mathscr{D}$ representa a la diagonal y $D(m)$ es la diagonal de m
4. $\sigma(x_{i_1}, \ldots, x_{i_n}, \overline{D(m)}))$
   Modus ponens sobre (2) y (3)
5. $(\alpha_{D(m)} \Rightarrow \sigma(x_{i_1}, \ldots, x_{i_n}, \overline{D(m)}))$
   Teorema de la deducción

y para la contraparte:

1. $\sigma(x_{i_1}, \ldots, x_{i_n}, \overline{D(m)})$
   Hipótesis.
2. $\mathscr{D}(\overline{m}, x)$
   Hipótesis.
3. $(\mathscr{D}(\overline{m}, x) \Rightarrow x = \overline{D(m)})$
   Unicidad en la representación de la diagonal.
4. $x = \overline{D(m)}$
   MP entre (2) y (3).
5. $\sigma(x_{i_1}, \ldots, x_{i_n}, \overline{D(m)})$
   Por sustitución de iguales en (1).
6. $(\mathscr{D}(\overline{m}, x) \Rightarrow \sigma(x_{i_1}, \ldots, x_{i_n}, x))$
   Teorema de la deducción sobre (2) y (5).
7. $(\forall x(\mathscr{D}(\overline{m}, x) \Rightarrow \sigma(x_{i_1}, \ldots, x_{i_n}, x)))$
   Generalización.
8. $\sigma(x_{i_1}, \ldots, x_{i_n}, \overline{D(m)}) \Rightarrow \alpha_{D(m)}$
   Teorema de la deducción.

Haciendo $e = D(m)$ tenemos que

$$AP \vdash (\sigma(x_{i_1}, \ldots, x_{i_n}, \overline{e}) \Leftrightarrow \alpha_e(x_{i_1}, \ldots, x_{i_n}))$$

□

La fórmula $\alpha_e(x_{i_1}, \ldots, x_{i_n})$ expresa algo acerca de su propio código y, como esto se puede hacer para cualquier fórmula σ, el Teorema del Punto Fijo se podría enunciar de la siguiente manera: 'Para cualquier propiedad que se pueda expresar en la aritmética formal, existe una fórmula que expresa la propiedad sobre sí misma'.

Capítulo 5

Problemas de Clasificación

Durante toda la tesis se ha ido construyendo en paralelo la teoría de las Máquinas de Turing y la teoría de la Aritmética Formal. En este capítulo se termina con dos resultados importantes: el Teorema de Rice, que trata con la teoría de la computabilidad, y un análogo, que trata con la aritmética. Ambos representan problemas para cualquier clasificación computable de los algoritmos y de las afirmaciones aritméticas, respectivamente.

De fondo, ambos teoremas están basados en la autorreferencia que se mostró inevitable en el capítulo anterior.

5.1. Teorema de Rice

Teorema 3. *(Teorema de Rice [1953])*
Para cualquier partición recursiva, no trivial, de los números naturales ($p : \mathbb{N} \to \{0,1\}$)*, existe un par de naturales* n *y* m *de forma que* $p(n) \neq p(m)$ *y* $\mathscr{M}_n \equiv \mathscr{M}_m$.

Demostración. Sea p partición recursiva no trivial y sean n y m tales que $p(n) = 0$ y $p(m) = 1$. Defínase a la siguiente función:

$$\psi(x,y) = \begin{cases} \mathscr{M}_n(x) & \text{si } p(y) = 1 \\ \mathscr{M}_m(x) & \text{si } p(y) = 0 \end{cases}$$

la función $\psi(x,y)$ es igual a $\mathscr{M}_n(x) \cdot p(y) + \mathscr{M}_m(x) \cdot (1 - p(y))$, por lo que es recursiva y, por lo tanto, computable. Aplicando el Teorema de Recursión se obtiene un número e tal que $\psi(x,e) = \mathscr{M}_e(x)$.
Si $p(e) = 0$, entonces $\mathscr{M}_e(x) = \psi(x,e) = \mathscr{M}_m(x)$, pero $p(e) \neq p(m)$.
Análogamente, si $p(e) = 1$, entonces $\mathscr{M}_e(x) = \mathscr{M}_n(x)$, pero $p(e) \neq p(n)$. □

5.2. Análogo al Teorema de Rice, para la aritmética

Teorema 4. *(Análogo al teorema de Rice)*
Para cualquier partición recursiva de los números naturales ($p : \mathbb{N} \to \{0,1\}$), no trivial sobre los números de fórmulas, existe un par de naturales n y m de forma que $p(n) = 0$, $p(m) = 1$ y $AP \vdash (\alpha_n \Leftrightarrow \alpha_m)$.

Demostración. Sea p partición recursiva no trivial sobre los números de fórmulas y sea $\mathscr{P}$ representación de ésta. Sean n y m tales que $p(n) = 0$ y $p(m) = 1$. Defínase la siguiente fórmula $\Psi(x)$:

$$((\mathscr{P}(x, 0) \Rightarrow \alpha_m) \wedge (\mathscr{P}(x, \overline{1}) \Rightarrow \alpha_n))$$

Aplicando el Teorema del Punto Fijo se obtiene un número e tal que

$$AP \vdash (\alpha_e \Leftrightarrow \Psi(e)) \tag{5.1}$$

Si $p(e) = 0$ entonces, por la representabilidad de p, se tiene que

$$AP \vdash \mathscr{P}(e, 0) \tag{5.2}$$

A continuación se muestra que $AP \vdash (\alpha_e \Leftrightarrow \alpha_m)$:

1. α_e (Hipótesis)
2. $((\mathscr{P}(e, 0) \Rightarrow \alpha_m) \wedge (\mathscr{P}(e, \overline{1}) \Rightarrow \alpha_n))$ (Modus ponens entre 5.1 y 1)
3. α_m (Modus ponens entre 5.2 y la primera parte de la conjunción 2)
4. $(\alpha_e \Rightarrow \alpha_m)$ (Teorema de la deducción)

y para la contraparte:

5. α_m (Hipótesis)
6. $(\mathscr{P}(e, 0) \Rightarrow \alpha_m)$ (De la tautología $A \Rightarrow (B \Rightarrow A)$ y modus ponens con 5)
7. $\mathscr{P}(e, \overline{1})$ (Hipótesis para hacer prueba por contradicción)
8. $((\mathscr{P}(e, 0) \wedge \mathscr{P}(e, \overline{1})) \Rightarrow 0 = \overline{1})$ (Porque $\mathscr{P}$ representa a p)

9. $(0 = \overline{1} \wedge (\neg 0 = \overline{1}))$ (Conjunción del primero que se obtiene por modus ponens de la conjunción de 5.2 y 7, y el segundo que se obtiene porque 0 no es sucesor de nadie y 1 sí)
10. $(\neg \mathscr{P}(e, \overline{1}))$ (Prueba por contradicción)
11. $(\mathscr{P}(e, \overline{1}) \Rightarrow \alpha_n)$ (De la tautología $\neg A \Rightarrow (A \Rightarrow B)$ y modus ponens con el anterior)
12. $((\mathscr{P}(e, 0) \Rightarrow \alpha_m) \wedge (\mathscr{P}(e, \overline{1}) \Rightarrow \alpha_n))$ (Conjunción de 6 y 11)
13. $(\alpha_m \Rightarrow \Psi(e))$ (Teorema de la deducción)
14. $(\alpha_m \Rightarrow \alpha_e)$ (Transitividad de la implicación con 5.1 y 13)

De esta manera se muestra que $AP \vdash (\alpha_e \Leftrightarrow \alpha_m)$, pero $p(e) \neq p(m)$ Análogamente, si $p(e) = 1$, se concluye que $AP \vdash (\alpha_e \Leftrightarrow \alpha_n)$, pero $p(e) \neq p(n)$. □

Nótese que, a pesar de que el teorema se enuncia para fórmulas en general, éste se puede generalizar a cualquier conjunto A de fórmulas que sea *cerrado bajo fórmulas* Ψ, es decir, que para cualesquiera dos fórmulas de A y cualquier partición p, la fórmula Ψ formada con ellas dos y su punto fijo también estén en A. Por ejemplo, el conjunto de enunciados es *cerrado bajo fórmulas* Ψ. El conjunto de fórmulas que tienen exactamente una variable libre x_i también lo es.

También es importante notar que, aunque ambos teoremas se enuncian para particiones recursivas, la clase de las funciones computables es idéntica a la clase de las funciones recursivas, por lo que los teoremas sí representan algo importante para la teoría de la computabilidad. A continuación se verán algunas aplicaciones, posiblemente relevantes en la teoría de la computabilidad, de ambos teoremas.

5.3. Problemas de clasificación de algoritmos

1. Para cualquier función f, considérese la siguiente función:

$$Eq_f(x) = \begin{cases} 1 & \text{si } \mathscr{M}_x \text{ computa } f \\ 0 & \text{en caso contrario} \end{cases}$$

Esta función no es computable.

Demostración. Si lo fuera, dicha partición estaría en contradicción con el teorema de Rice. □

2. Considérese a la siguiente función:

$$f(x,y) = \begin{cases} 1 & \text{si } \mathscr{M}_x \equiv \mathscr{M}_y \\ 0 & \text{si } \mathscr{M}_x \not\equiv \mathscr{M}_y \end{cases}$$

Esta función no es computable.

Demostración. Para cualquier natural d, $p(x) = f(x,d)$ contradice al teorema de Rice. □

3. Considérese la siguiente función, donde el segundo argumento corre sobre las descripciones de cinta:

$$\Delta(x,y) = \begin{cases} 1 & \text{si } \mathscr{M}_x \text{ llega a } q_f \text{ con entrada } y \\ 0 & \text{si } \mathscr{M}_x \text{ nunca se detiene con entrada } y \end{cases}$$

Esta función no es computable (Problema de la detención).

Demostración. Para cualquier natural d, $p(x) = \Delta(x,d)$ contradice al teorema de Rice. □

4. Para cada natural n considérese a la función:

$$Im_n(x) = \begin{cases} 1 & \text{si } n \text{ está en la imágen de la función computada por } \mathscr{M}_x \\ 0 & \text{en caso contrario} \end{cases}$$

Esta función no es computable.

Demostración. Si lo fuera, estaría en contradicción con el teorema de Rice. □

5. Para cada natural n considérese a la función:

$$Dom_n(x) = \begin{cases} 1 & \text{si } n \text{ está en el dominio de la función computada por } \mathscr{M}_x \\ 0 & \text{en caso contrario} \end{cases}$$

Esta función no es computable.

Demostración. Si lo fuera, estaría en contradicción con el teorema de Rice. □

5.4. Problemas de clasificación de fórmulas

1. Para cualquier fórmula α, considérese la siguiente función:

$$Eq_\alpha(x) = \begin{cases} 1 & \text{si } AP \vdash (\alpha_x \Leftrightarrow \alpha) \\ 0 & \text{en caso contrario} \end{cases}$$

Esta función no es computable.

Demostración. Si lo fuera, dicha partición estaría en contradicción con el análogo al teorema de Rice. □

2. Considérese a la siguiente función:

$$f(x,y) = \begin{cases} 1 & \text{si } AP \vdash (\alpha_x \Leftrightarrow \alpha_y) \\ 0 & \text{en caso contrario} \end{cases}$$

Esta función no es computable.

Demostración. Para cualquier natural d, $p(x) = f(x,d)$ contradice al análogo al teorema de Rice. □

3. Considérese a la función:

$$Teo(x) = \begin{cases} 1 & \text{si } AP \vdash \alpha_x \\ 0 & \text{en caso contrario} \end{cases}$$

Esta función no es computable (Teorema de Church [1936]).

Demostración. Si fuera computable estaría en contradicción con el análogo al teorema de Rice. □

4. Considérese a todas las fórmulas que tienen exactamente una variable libre x_i. Para cada n natural, considérese a la siguiente función:

$$Prop_n(x) = \begin{cases} 1 & \text{si } AP \vdash \alpha_x(x_i|\overline{n}) \\ 0 & \text{en caso contrario} \end{cases}$$

Esta función no es computable (Las propiedades del natural n no es un conjunto recursivo).

Demostración. El conjunto de fórmulas con exactamente una variable libre x_i es cerrado bajo fórmulas Ψ, así que ésta función estaría en contradicción con el análogo al teorema de Rice, de ser computable. □

5. Cualquier teoría T, con un conjunto recursivo de axiomas propios (o sea, que su función característica sea recursiva), donde las funciones recursivas sean representables y se demuestre $0 \neq 1$, es inconsistente o incompleta (Teorema de Gödel [1931])[1].

Demostración. Si el conjunto de axiomas propios es recursivo, la propiedad $Dem_T(x)$[2] es recursiva, y por lo tanto $Ded_T(n)$, la enumeración de las consecuencias de T, es recursiva. Considérese la siguiente función:

$$p(x) = \begin{cases} 1 & \text{si } AP \vdash \alpha_x \\ 0 & \text{si } AP \vdash (\neg\alpha_x) \end{cases}$$

Dicha función se puede computar haciendo uso de Ded_T así:

$$p(x) = \mu(sg[\mu_y(Ded_T(y) = x) + 1] \vee \overline{sg}[\mu_y(Ded_T(y) = \sharp[(\neg] * x * 2^{\sharp)}) + 1])$$

En caso de que T fuera completa, la función $p(x)$ sería una partición con el dominio completo de los enunciados. Como el conjunto de enunciados es cerrado bajo fórmulas Ψ, se puede aplicar el análogo al teorema de Rice. Aplicándolo resulta que existen n y m tales que $AP \vdash (\alpha_n \Leftrightarrow \alpha_m)$, pero para α_n existe una demostración y para α_m existe una demostración de su negación. Por modus ponens se tiene que $AP \vdash \alpha_m$. De esta manera, se tiene que $AP \vdash (\alpha_m \wedge (\neg\alpha_m))$, por lo que T es inconsistente. □

6. Cualquier teoría T con las condiciones anteriores y consistente tendrá enunciados que no se demuestran ni se refutan. Estos se llaman *indecidibles*. Sería conveniente poder clasificar a los indecidibles por clases de equivalencias, pero esto es claramente imposible. Peor aún, también es imposible separar a aquellos de los decidibles.

Demostración. Aplicando el análogo al teorema de Rice y el hecho de que no puede haber un decidible y un indecidible equivalentes entre sí. □

[1] De hecho, Gödel lo enunció de una manera más débil, pidiendo ω-consistencia en lugar de consistencia. Aquí se demuestra la versión más general que Rosser probó en 1936.

[2] $Dem_T(x)$ es la propiedad 'ser una demostración formal en T'.

7. Si T es una teoría que cumple las hipótesis anteriores, no existe una *forma canónica*, para cada clase de equivalencias (módulo la demostración en T) de fórmulas, tal que exista un algoritmo que traduzca cualquier fórmula a su forma canónica, a menos que la teoría sea inconsistente y todas las fórmulas pertenezcan a una sola clase de equivalencia. Esto se puede enunciar como que 'no existe una máquina que entienda la semántica de cualquier expresión'.

Demostración. Sea k el número de un teorema de T. Si existiera un algoritmo para computar dicha función $FormCan$, se tendría a la función computable

$$p(x) = \overline{sg}[(FormCan(x) \dot{-} FormCan(k)) + (FormCan(k) \dot{-} FormCan(x))]$$

que separa a los teoremas del resto de las fórmulas (pues una fórmula es teorema si y sólo si es equivalente a un teorema distinto). Esto está en contradicción con el teorema de Church. □

Aunque ya se mencionó en los ejemplos, no está de más aclarar que aunque todo el trabajo se hizo sobre las Máquinas de Turing y sobre la Aritmética de Peano, los problemas de clasificación son aplicables a cualquier formalización recursiva de los algoritmos o a cualquier teoría formal recursiva, con tal de que los objetos de la primera sean capaces de computar todas las funciones recursivas y en la segunda se representen a todas las funciones recursivas.

Conclusiones

El paralelismo entre los problemas de clasificación de algoritmos y los problemas de clasificación de fórmulas es evidente. En ambos casos, las limitaciones son consecuencia de la autorreferencia que los Teoremas Diagonales demuestran ser inevitable. Pero, más allá de los Teoremas Diagonales, está el poder representacional de los números naturales. Irónicamente, es gracias a su poder intrínseco que cualquier sistema formal que los maneje (los opere y los relacione) presentará problemas; sencillamente, porque los números pueden representar al mismo sistema formal.

Los números son la herramienta que se usó en esta tesis para representar a la *manipulación simbólica*, misma que se usa para hacer la teoría formal de los números (AP), y que se formaliza como se hizo aquí (MT). Sin embargo, el problema no está en los números; estos sólo son el medio. La forma en la que se obtuvieron los resultados nos enseña que cualquier ciencia suficientemente fuerte para representar a la manipulación simbólica se podrá estudiar a sí misma. Éste es el problema.

El trabajo comenzó después de escuchar una plática de Carlos Torres Alcaraz en el Congreso Nacional de Matemáticas de la Sociedad Matemática Mexicana, celebrado en Valle de Bravo en el año 2008. En ésta se presentó una prueba del Teorema de Gödel a partir del Teorema de Rice (un problema de decidibilidad en la aritmética deducido a partir de un problema de decidibilidad en la teoría de la computabilidad), basada en el paralelismo de las dos pruebas (la aritmetización). Carlos Torres propuso el reto de demostrar el Teorema de Rice a partir del Teorema de Gödel y, nosotros, en busca de un tema para tesis, lo tomamos; más que nada, como excusa para empezar a estudiar algo sobre lo cuál se pudiera escribir una tesis.

El reto nunca concluyó pero los resultados fueron más de los esperados pues, aunque el paralelismo siempre estuvo ahí y no se esconde de nadie, su profundidad es mayor que lo que a simple vista notamos. En primer lugar,

no esperábamos tal similitud entre los Teoremas Diagonales y, en segundo lugar, el Análogo al Teorema de Rice, que enunciamos sólo una vez que ya estaba muy estudiado el paralelismo, sólo era una conjetura y no tenía por qué salir. Por ningún lado encontramos una referencia a dicha afirmación pero afortunadamente nos cayó del árbol una demostración para ella. Con la certeza de este resultado y sus consecuencias, el paralelismo quedó más fuerte que nunca.

Bibliografía

[1] Elliott Mendelson, *Introduction to Mathematical Logic*, Fourth Edition, Chapman & Hall (1997)

[2] Douglas S. Bridges, *Computability -A Matematical Sketchbook*, Springer-Verlag (1994)

[3] V.A. Uspenskĭ, *Máquina de Post*, Editorial MIR (1979)

[4] John Barkley Rosser, *Extensions of some theorems of Gödel and Church*, reprinted from the Journal of Symbolic Logic vol. 1 (1936)

[5] Stephen Kleene, *On notation for ordinal numbers*, The Journal of Symbolic Logic, 3 (1938)

[6] Henry Gordon Rice, *Classes of Recursively Enumerable Sets and Their Decision Problems*, Trans. Amer. Math. Soc. (1953)

[7] Kurt Gödel, *Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I.*, Monatshefte für Mathematik und Physik 38 (1931), en Solomon Feferman, *Collected Works of Kurt Gödel*, Oxford University Press (1990).