Tesis de Licenciatura

Una nueva versión de la probabilidad de detención

Sergio Daicz
sdaicz@dc.uba.ar

Directora: Dra. Verónica Becher

Supervisor: Dr. Gregory Chaitin

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Septiembre de 2000

Resumen

El trabajo de Martin-Löf, Chaitin, Kolmogorov, Solovay y otros permitió tener una definición matemática de número real aleatorio. Chaitin encontró un ejemplo natural de número aleatorio: Ω, la probabilidad de detención de una máquina universal autodelimitante (también llamada máquina universal de Chaitin, MUC). Además de ser aleatorio, este número tiene la propiedad de ser aproximable en el límite desde abajo, propiedad que en la literatura se denomina real recursivamente enumerable (r. e.).

Recientemente Slaman, a partir del trabajo de Calude y otros, ha demostrado que la clase de reales aleatorios r. e. coincide con la de los Ω-numbers, es decir que todo real aleatorio r. e. es la probabilidad de detención de alguna máquina universal autodelimitante.

Surge entonces el interés por estudiar reales aleatorios fuera de esta clase. En este trabajo demostraremos que *la probabilidad de que una MUC lea sólo una porción finita de su cinta de programa, aun disponiendo de tiempo infinito*, es un número aleatorio no r. e. Veremos que es totalmente impredecible, aun disponiendo de un oráculo para el problema de la detención. Esto puede resultar de interés en el contexto de una teoría de complejidad algorítmica de conjuntos.

Índice general

Capítulo 1

Introducción

El trabajo de Martin-Löf [12], Chaitin [3], Kolmogorov [10], Solovay y otros permitió tener una definición matemática de número real aleatorio. Chaitin [4] encontró un ejemplo natural de número aleatorio: Ω, la probabilidad de detención de una máquina universal autodelimitante (también llamada máquina universal de Chaitin, MUC). Además de ser aleatorio, este número tiene la propiedad de ser aproximable en el límite desde abajo, propiedad que en la literatura se denomina real recursivamente enumerable (r. e.).

Recientemente Slaman [13], a partir del trabajo de Calude y otros [2], ha demostrado que la clase de reales aleatorios r. e. coincide con la de los Ω-numbers, es decir que todo real aleatorio r. e. es la probabilidad de detención de alguna máquina universal autodelimitante.

Surge entonces el interés por estudiar reales aleatorios fuera de esta clase. En este trabajo demostraremos que *la probabilidad de que una MUC lea sólo una porción finita de su cinta de programa, aun disponiendo de tiempo infinito*, es un número aleatorio no r. e. Veremos que es totalmente impredecible, aun disponiendo de un oráculo para el problema de la detención. Esto puede resultar de interés en el contexto de una teoría de complejidad algorítmica de conjuntos, como la que inicia Chaitin en [5].

El trabajo se organiza de la siguiente manera. En el capítulo 2 presentamos las definiciones de máquinas autodelimitantes para cómputos finitos e infinitos, la noción de complejidad algorítmica, de número aleatorio, de número r. e. y de computabilidad relativa, así como los resultados previos que necesitaremos a continuación. En el capítulo 3 demostramos nuestro resultado. Finalmente, en el capítulo 4 proponemos posibles trabajos siguiendo esta línea de investigación.

Capítulo 2

Definiciones y trabajo previo

2.1. Cadenas, secuencias y probabilidades

$\Sigma = \{0, 1\}$ es el alfabeto binario. Σ^* es el conjunto de *cadenas* binarias (finitas). $|s|$ es la longitud de la cadena s. λ es la cadena vacía. Σ^+ es el conjunto de cadenas binarias no vacías. Σ^n es el conjunto de cadenas binarias de longitud exactamente n. a, b, etc. representarán cadenas o números enteros, según el contexto. A, B, etc., representarán conjuntos (finitos o infinitos) de cadenas. Se puede definir un orden sobre Σ^* considerando las cadenas primero por su longitud y luego lexicográficamente, es decir: $\lambda < 0 < 1 < 00 < 01 < 10 < 11 < 000 \ldots$. Definimos $string_i$ como la i-ésima cadena en ese orden, o sea $string_1 = \lambda$, $string_2 = 0$, $string_6 = 10$, etc. Σ^ω es el conjunto de las *secuencias* binarias (infinitas). $\mathbf{x}$, $\mathbf{y}$, etc., representarán secuencias o números reales. x_i es el dígito en la posición i de la secuencia $\mathbf{x}$, donde la primer posición es la 1. $\mathbf{x}(i)$ es la cadena formada por los primeros i dígitos de $\mathbf{x}$, o sea $\mathbf{x}(i) = x_1 x_2 \ldots x_i$. $\mathbf{X}$, $\mathbf{Y}$, etc., representarán conjuntos (finitos o infinitos) de secuencias. sA es el conjunto $\{sa | a \in A\}$. También usaremos letras griegas, como α y Ω para referirnos a ciertas secuencias y números reales en particular.

Podemos pensar un proceso aleatorio de elección de una cadena de la siguiente manera. Se asigna igual probabilidad al 0 y al 1, y se obtiene cada dígito de la cadena de manera independiente. De esta manera, si elegimos una cadena de n dígitos, tenemos 2^n posibles cadenas, y la probabilidad de obtener una cadena determinada es 2^{-n}.

Las secuencias binarias infinitas se pueden poner en correspondencia con los números reales en el intervalo $[0, 1]$. Para hacer esto vemos a las secuencias como la representación binaria de un número real. Es decir, identificamos la secuencia $\mathbf{x}$ con el número real:

$$\sum_{i=1}^{\infty} x_i 2^{-i}$$

Todo número real en $[0, 1]$ tiene entonces una secuencia correspondiente. Los racionales diádicos —de la forma $k2^{-i}$, con k e i enteros— se corresponden con dos secuencias, una terminada en infinitos 0s y otra terminada en infinitos 1s. Pero como son un conjunto numerable esto no afectará a las consideraciones sobre probabilidades que haremos más adelante. Como convención, siempre elegiremos la representación con infinitos 1s. Abusaremos del lenguaje y frecuentemente nos referiremos a un número real como si fuera una secuencia binaria y viceversa, teniendo en cuenta esta correspondencia. En algunas ocasiones, también nos referiremos a una cadena a como si fuera un número real. En este caso se trata del número:

$$\sum_{i=1}^{|a|} a_i 2^{-i}$$

Si mediante el proceso aleatorio descripto más arriba se elige una secuencia infinita, obtenemos la distribución uniforme de los números reales en el intervalo $[0, 1]$. La probabilidad de obtener un número real en particular es 0. Pero escribiremos $\mu(\mathbf{X})$ para indicar la probabilidad de que la secuencia elegida pertenezca a un conjunto de secuencias $\mathbf{X}$. Por ejemplo, si tomamos el conjunto $\mathbf{X} = s\Sigma^\omega$, es decir el conjunto de todas las secuencias que comienzan con la cadena s, la probabilidad de que una secuencia pertenezca a ese conjunto es $2^{-|s|}$, por lo que $\mu(\mathbf{X}) = 2^{-|s|}$.

Decimos que un conjunto de cadenas A es *libre de prefijos* si ningún elemento de A es prefijo propio de otro elemento de A, es decir, si para todo $a, b \in A$, y para todo $s \in \Sigma^+$: $as \neq b$. La *desigualdad de Kraft* [11] dice que si A es libre de prefijos entonces la serie:

$$\sum_{a \in A} 2^{-|a|}$$

converge y su límite es un número real en el intervalo $[0, 1]$. Escribiremos 2^{-A} para referirnos a ese número.

Podemos hacer ahora la siguiente observación, que nos permitirá transformar probabilidades sobre conjuntos de secuencias en sumas de aportes relativos a longitudes de cadenas:

Observación 2.1 *Dado un conjunto de cadenas A libre de prefijos, la probabilidad de que una secuencia elegida mediante la distribución uniforme tenga como prefijo una cadena de A es 2^{-A}. Es decir: $\mu(A\Sigma^\omega) = 2^{-A}$*

2.2. Máquinas autodelimitantes

Las máquinas autodelimitantes fueron introducidas por Chaitin en [4] para desarrollar su teoría de complejidad algorítmica de cadenas finitas. En

[5] Chaitin extiende su trabajo a cómputos infinitos que enumeran conjuntos, utilizando un modelo de máquina ligeramente diferente. En [6] presenta una visión más integrada de ambas versiones, y es la que seguimos en esta sección.

Una *máquina autodelimitante* es una máquina de Turing con una cinta de programa, una cinta de trabajo y una cinta de salida. La cinta de programa es de sólo lectura, y la de salida es de sólo escritura. Ambas son infinitas hacia la derecha y sus cabezas solo se pueden mover en esa dirección. La cinta de trabajo es de lectura y escritura, es infinita en ambas direcciones y su cabeza se puede mover en ambas direcciones.

Al inicio de cada ciclo, la máquina puede estar en uno de una cantidad finita de estados posibles. Durante el ciclo la máquina puede realizar alguna de las siguientes acciones: mover la cabeza de la cinta de programa una posición a la derecha, escribir un 0, un 1, o un blanco en la celda actual de la cinta de trabajo, mover la cabeza de la cinta de trabajo una posición a la derecha o a la izquierda, o escribir un 0 o un 1 en la celda actual de la cinta de salida y mover la cabeza de la misma una posición a la derecha. Luego de efectuar la accion la máquina cambia de estado y comienza un nuevo ciclo. La acción a realizar y el próximo estado son función del estado actual y de los símbolos en las celdas actuales de las cintas de programa y de trabajo. Por lo tanto, una máquina se puede describir indicando el conjunto de estados posibles, uno de los cuales será llamado *estado inicial*, más un conjunto finito de tuplas $< estado_1, simbolo_p, simbolo_t, accion, estado_2 >$, en el que exista a lo sumo una tupla por cada posible combinación de las tres primeras componentes. Cada tupla se interpreta de la siguiente manera: "Si la máquina está en $estado_1$, y en la cinta de programa se lee $simbolo_p$, y en la cinta de trabajo se lee $simbolo_t$, entonces realizar *accion* y pasar a $estado_2$". Si durante el cómputo la máquina llega a una combinación de $estado_1$, $simbolo_p$ y $simbolo_t$ para la cual no hay ninguna tupla en la tabla decimos que la máquina se *detiene*, y de allí en adelante permanecerá en esa configuración.

Sean p una cadena y M una máquina autodelimitante. Un cómputo para $M(p)$ comienza de la siguiente manera. En la cinta de programa hay una secuencia perteneciente a $p\Sigma^\omega$. La cabeza de la cinta de programa se encuentra en el extremo izquierdo de la misma. La cinta de trabajo está en blanco. La cabeza de la cinta de salida está en el extremo izquierdo de la misma. La máquina se encuentra en el estado inicial.

Esta definición de máquina nos servirá para hablar tanto de cómputos finitos como infinitos. El resultado de un cómputo finito es una cadena finita. El resultado de un cómputo infinito es un conjunto, posiblemente infinito, de números naturales.

Definición 2.2 (Chaitin [6]) Decimos que p es un *programa válido* para M si al iniciar M de la manera recién descripta, ésta se detiene luego de una cantidad finita de ciclos con la cabeza de la cinta de programa apuntando

al último bit de p.

En ese caso decimos también que $M(p)$ está definido y escribimos $M(p) \downarrow$. En caso contrario escribimos $M(p) \uparrow$. Si está definido, el resultado de $M(p)$ es la cadena que se encuentra en la cinta de salida a la izquierda de la cabeza de la misma. Escribiremos también $M(\mathbf{x}) \downarrow$ si existe algún prefijo p de $\mathbf{x}$ tal que $M(p) \downarrow$.

Definición 2.3 Decimos que p es un *programa válido para un cómputo infinito* en M si al iniciar M de la misma manera, la cabeza de la cinta de programa llega en algún momento a apuntar al último bit de p, y luego no se mueve nunca hacia la derecha.

En ese caso decimos también que $M^\infty(p)$ está definido y escribimos $M^\infty(p) \downarrow$. En caso contrario escribimos $M^\infty(p) \uparrow$. En caso de estar definido, el resultado del cómputo se interpreta de la siguiente manera: n está en el conjunto $M^\infty(p)$ si y sólo si en algún momento del cómputo aparece en la cinta de salida a la izquierda de la cabeza escritora una cadena de exactamente n 0s delimitada por 1s. Escribiremos $M^\infty(\mathbf{x}) \downarrow$ si existe algún prefijo p de $\mathbf{x}$ tal que $M^\infty(p) \downarrow$.

Definición 2.4

1. K_M es el conjunto de programas válidos para la máquina M: $K_M = \{p | M(p) \downarrow\}$
2. K_M^∞ es el conjunto de programás válidos para cómputos infinitos en M^∞: $K_M^\infty = \{p | M^\infty(p) \downarrow\}$

Se puede ver ahora por qué llamamos autodelimitante a la máquina recién descripta. Como no puede haber blancos en la cinta de programa, ni ninguna otra manera externa de delimitación, un programa debe contener dentro de sí mismo la información necesaria para saber dónde termina:

Proposición 2.5 (Chaitin [4]) *Para toda máquina autodelimitante M, el conjunto K_M es libre de prefijos.*

Demostración. Sea $a \in \Sigma^*$ tal que $M(a) \downarrow$. Sea $t \in \Sigma^+$. Si se intenta ejecutar el cómputo $M(at)$, la máquina se detendrá con la cabeza de la cinta de programa apuntando al último bit de a, por lo que nunca podrá llegar a apuntar al último bit de at. Por lo tanto debe ser $at \notin \{p | M(p) \downarrow\}$, por lo que este último conjunto es libre de prefijos. □

Un razonamiento similar se puede seguir para ver que el conjunto K_M^∞ es libre de prefijos.

Elegiremos ahora una máquina autodelimitante U con la capacidad de simular cualquier otra máquina:

Definición 2.6 (Chaitin [4]) Supongamos fija una numeración efectiva cualquiera de todas las máquinas autodelimitantes, de manera que M_i es

la i-ésima máquina en dicha numeración. U lee su cinta de programa hasta encontrar el primer 1. Si la cantidad de 0s leídos es i, comienza a simular la ejecución de M_i, tomando el resto de la cinta de programa como el programa para M_i. Llamaremos a U *máquina universal de Chaitin.*

2.3. Complejidad Algorítmica

Intuitivamente, la complejidad de un objeto es la mínima cantidad de información necesaria para reconstruirlo, es decir, la longitud de la descripción más corta posible de dicho objeto. Pero para hacer precisa esta definición se debe especificar cuál es el mecanismo que traduce descripciones en objetos. En la teoría de información algorítmica desarrollada por Chaitin en [4], este mecanismo será una máquina autodelimitante.

Entonces, podemos formalizar la definición diciendo que la complejidad de una cadena s es la longitud del programa más corto que en una máquina autodelimitante devuelve s como resultado:

Definición 2.7 (Chaitin [4]) La *complejidad algorítmica* de la cadena s con respecto a la máquina autodelimitante M es $H_M(s) = \text{mín}\{|p| \ |M(p) = s\}$.

Entonces, la complejidad de una cadena depende del "mecanismo de traducción", es decir, de la máquina particular que se haya elegido para medir complejidades. Pero el siguiente corolario de las definiciones de complejidad algorítmica y máquina universal nos muestra que de alguna manera podemos tener una definición "absoluta" de complejidad:

Corolario 2.8 (Chaitin [4]) *Sea U una máquina universal de Chaitin. Para toda máquina autodelimitante M existe una constante $sim(M)$ tal que para toda cadena s:*

$$H_U(s) \leq H_M(s) + sim(M)$$

Por lo tanto, para cualquier par de máquinas universales U_1 y U_2 hay una constante c tal que para toda cadena s:

$$|H_{U_1}(s) - H_{U_2}(s)| \leq c$$

A este resultado se lo conoce como *teorema de invarianza*, y nos dice que no importa qué máquina universal se use: la complejidad de las cadenas no podrá diferir mucho según sea medida con una o con otra. En este sentido podemos pensar que la complejidad de una cadena es un atributo inherente a la misma, independiente del sistema de codificación.

De aquí en adelante, fijaremos una máquina universal de Chaitin particular U. Escribiremos K por K_U, K^∞ por K_U^∞, y $H(s)$ por $H_U(s)$.

2.4. Reales aleatorios

Presentaremos ahora la definición de número real aleatorio. Para ello definiremos qué es una secuencia aleatoria y diremos que un número real en el intervalo $[0, 1]$ es aleatorio si su secuencia binaria correspondiente lo es. Si bien esta definición se da en función de la representación en base 2, se puede ver que es invariante con respecto a la base elegida. Es decir que la propiedad de ser aleatorio es un atributo del número y no depende del sistema elegido para representarlo (cf. Calude [1, Sec. 6.7]).

2.4.1. Definición de Martin-Löf

(El material de esta sección está basado en [1, Sec. 6.3] y [13]).

Una manera en que uno podría tratar de definir la noción de secuencia aleatoria es pidiendo que no tenga ninguna propiedad "excepcional". Esto se puede formalizar diciendo que una propiedad es un conjunto $\mathbf{P}$ de secuencias, y diciendo que $\mathbf{P}$ es excepcional si la probabilidad de que una secuencia elegida mediante la distribución uniforme pertenezca a $\mathbf{P}$ es 0 (es decir, si $\mu(\mathbf{P}) = 0$). Entonces uno podría querer decir que una secuencia es aleatoria si cumple con todas las propiedades que son verdaderas para casi todas las secuencias, es decir, si pertenece a todas las $\mathbf{P}$ tales que $\mu(\mathbf{P}) = 1$. Sin embargo, para cada secuencia $\mathbf{x}$ es posible definir una propiedad $\mathbf{P_x}$ de manera que $\mathbf{x}$ no pertenece a $\mathbf{P_x}$ y $\mu(\mathbf{P_x}) = 1$, por ejemplo definiendo:

> $\mathbf{y}$ está en $\mathbf{P_x}$ si y sólo si para todo $n \geq 1$ existe un $m \geq n$ tal que $x_m \neq y_m$.

Por lo tanto, ninguna secuencia cumple con la definición de aleatoriedad propuesta. La solución a este problema es no pedir que la secuencia evite todas las posibles propiedades de probabilidad 0 sino sólo las que son definibles mediante un algoritmo:

Definición 2.9 (Martin-Löf [12]) Un test de aleatoriedad de Martin-Löf es una sucesión recursivamente enumerable $(A_n : n \geq 1)$ de subconjuntos de Σ^* tal que para todo n, $\mu(A_n\Sigma^\omega) \leq 2^{-n}$

La intersección de todos los $A_i\Sigma^\omega$ es un conjunto de secuencias de probabilidad 0, ya que está contenida en conjuntos de probabilidad arbitrariamente pequeña. Si una secuencia se encuentra en ese conjunto, entonces decimos que es *rechazada* por el test. En caso contrario es aceptada. Esta definición de test es lo suficientemente general como para incluir por ejemplo "todo test [de aleatoriedad] de uso presente o futuro en la estadística" [12]. Llegamos así a la definición de aleatoriedad de Martin-Löf. Una secuencia es aleatoria si pasa todo posible test efectivo de aleatoriedad:

Definición 2.10 (Martin-Löf [12]) Una secuencia $\mathbf{x}$ es Martin-Löf-aleatoria si y sólo si para todo test de Martin-Löf $(A_n : n \geq 1)$, $x \notin \bigcap_{n\geq 1} A_n\Sigma^\omega$.

2.4.2. Definición de Chaitin

Otra manera en que uno podría querer definir secuencia aleatoria es pidiendo que no tenga ningún tipo de regularidad, que sus bits sean totalmente impredecibles. El concepto de Chaitin de complejidad algorítmica permite formalizar esta noción, ya que cualquier regularidad en una secuencia podría ser aprovechada para comprimirla, es decir, para generar una cierta cantidad de bits de la misma a partir de un programa compuesto por una cantidad sustancialmente menor. La ausencia de regularidad se puede expresar entonces como incompresibilidad. Obtenemos así la definición de Chaitin de secuencia aleatoria, que pide que la complejidad algorítmica de cada prefijo de la secuencia sea del orden de su longitud:

Definición 2.11 (Chaitin [4]) Una secuencia $\mathbf{x}$ es Chaitin-aleatoria si existe una constante c tal que para todo n $H(\mathbf{x}(n)) > n - c$.

Schnorr (anunciado por Chaitin en [4], ver también [7, Cap. 7]) demostró que ambas definiciones de aleatoriedad son equivalentes:

Teorema 2.12 (Schnorr) *Para toda secuencia $\boldsymbol{x}$, $\boldsymbol{x}$ es Martin-Löf-aleatoria si y sólo si es Chaitin-aleatoria.*

La equivalencia entre ambas definiciones —obtenidas mediante la formalización de ideas relativamente diferentes— es un buen indicio de que éstas capturan satisfactoriamente la noción intuitiva de aleatoriedad. De aquí en adelante llamaremos aleatoria a una secuencia que cumpla con cualquiera de ambas definiciones equivalentes. Y número aleatorio a aquel cuya expansión binaria sea una secuencia aleatoria.

2.4.3. La probabilidad de detención

La probabilidad de detención de una máquina autodelimitante M es la probabilidad de que M se detenga si se inicializa el contenido de la cinta de programa con una secuencia elegida al azar con distribución uniforme. Es decir, la probabilidad de que una secuencia elegida al azar comience con un programa válido para M:

Definición 2.13 (Chaitin [4]) La *probabilidad de detención* de una máquina M es

$$\mu(K_M\Sigma^\omega) = 2^{-K_M} = \sum_{M(p)\downarrow} 2^{-|p|}$$

Chaitin descubrió que la probabilidad de detención de una máquina universal es un ejemplo natural de número real aleatorio:

Teorema 2.14 (Chaitin, [4]) *Sea $\Omega = 2^{-K}$. Ω es un número real aleatorio.*

La idea general de la demostración es que si se conocen los primeros n bits de Ω, entonces se puede resolver el problema de la detención para todos los programas de longitud menor o igual que n. Y si se conocen todos los programas de longitud menor o igual que n que se detienen, se puede encontrar una cadena s que no sea el resultado de ninguno de esos programas. La complejidad de esta cadena debe ser mayor que n ya que no hay ningún programa más corto que la genere. Entonces, si se conocen los primeros n bits de Ω se puede generar una cadena de complejidad mayor que n. Pero esto implica que la complejidad de los primeros n bits de Ω también debe ser mayor que n, ya que si hubiera un programa más corto que los generara se lo podría utilizar para generar s.

Entonces, la probabilidad de que una secuencia comience con una cadena perteneciente a K es un número aleatorio. Es importante remarcar que la propiedad de un conjunto de cadenas de tener como probabilidad un número aleatorio no es recursivamente invariante, como veremos con el siguiente ejemplo:

Proposición 2.15 *Sea* $B = \{0^{i-1}1 | string_i \in K\}$. *Sea* $\beta = 2^{-B}$. β *no es un número aleatorio.*

Demostración. Por su definición, el i-ésimo bit de β es 1 si y sólo si $string_i$ es un programa válido para U. Y como dijimos en la demostración del teorema 2.14, si tenemos los primeros n bits de Ω podemos resolver el problema de la detención para todos las cadenas de longitud menor o igual que n. Pero la cantidad de cadenas de ese largo es del orden de 2^{n+1}. Por lo tanto, a partir de n bits de Ω se pueden obtener 2^{n+1} bits de β. O sea que $H(\beta(n))$ es del orden de $\log n$. Por lo tanto, β no es aleatorio. □

Es fácil ver que K y B son recursivamente isomorfos, ya que ambos son 1-completos (ver [9, Sec. 7.4]). Sin embargo, 2^{-K} es un número aleatorio y 2^{-B} no lo es.

2.5. Reales recursivamente enumerables

Un número real β es *recursivamente enumerable* (r. e.) si existe una sucesión computable creciente de racionales que converge a β. Una definición equivalente, que usaremos luego, es que el conjunto $\{p \in \mathbf{Q} | p < \beta\}$ es r. e.

Entonces, todo real computable es tambien r. e., pero hay números r. e. que no son computables. Hay que notar que si bien la sucesión pedida en la definición debe ser computable, la velocidad de convergencia puede ser muy lenta, incluso no computable. Por ejemplo, Ω es un caso de número que es r. e. pero no puede ser computable ya que si lo fuera se podría resolver el problema de la detención.

En los últimos años hubo importantes avances en la comprensión de las propiedades de los números reales r. e., entre los que podemos citar los

trabajos de Calude y otros [2] y de Slaman [13]. Uno de los resultados más interesantes es el siguiente:

Teorema 2.16 (Slaman [13]) *Sea β un número real aleatorio recursivamente enumerable. β es la probabilidad de detención de una máquina universal de Chaitin.*

Entonces, la clase de los Ω-numbers, los números que son la probabilidad de detención de una máquina universal de Chaitin, coincide con la de los números aleatorios recursivamente enumerables.

2.6. Máquinas con oráculos

Nos interesa estudiar la aleatoriedad de la probabilidad de que un cómputo infinito esté definido. Pero a diferencia de K, el conjunto K^∞ no es recursivamente enumerable por una máquina universal común. Entonces en algunas partes de nuestro análisis trabajaremos con una máquina que sí pueda enumerar K^∞, una máquina con un oráculo para el problema de la detención. Definimos ahora qué es una máquina con oráculo (cf. [9, Cap. 9]):

Sea A un conjunto cualquiera de cadenas. Una *máquina con oráculo para* A es una como la descripta en la sección 2.2 pero permitiendo adicionalmente tuplas de la forma $< estado_1, simbolo_p, simbolo_t, estado_2, estado_3 >$, que se interpretan de la siguiente manera: "Si la máquina está en $estado_1$, y en la cinta de programa se lee $simbolo_p$, y en la cinta de trabajo se lee $simbolo_t$, entonces si la cadena que se encuentra a la derecha de la cabeza de la cinta de trabajo pertenece a A pasar a $estado_2$; si no pasar a $estado_3$".

Se obtiene así el concepto de computabilidad relativa. Una función que puede ser calculada por una máquina con oráculo para A se dice *A-computable*, y un conjunto que sea enumerable por una máquina del mismo tipo se dice *A-recursivamente enumerable*. La mayoría de los resultados de la teoría de la computabilidad se mantienen al pasar a máquinas con oráculos, en particular la existencia de máquinas universales. Utilizaremos la siguiente notación:

Definición 2.17

1. U^A será una máquina universal de Chaitin fija con oráculo para A.
2. $H^A(x)$ es la longitud del programa más corto para producir x en U^A:
$$H^A(x) = H_{U^A}(x)$$
3. A' (o *salto* de A) es el conjunto de programas que se detienen en U^A:
$$A' = K_{U^A} = \{p | U^A(p) \downarrow\}$$

4. Ω^A es la probabilidad de detención de U^A:

$$\Omega^A = 2^{-A'}$$

Presentaremos ahora un par de propiedades que nos serán útiles más adelante:

Proposición 2.18 (Chaitin, [5]) *Para todo conjunto de cadenas A existe una constante c tal que para toda cadena x, $H^A(x) \leq H(x) - c$.*

Demostración. Hay una máquina M_i^A que se comporta exactamente como U (simplemente no utiliza el oráculo). Por lo tanto vale para toda cadena p que $U(p) = U^A(0^i 1 p)$. Por lo tanto, si $H(x)$ es la longitud del programa más corto que produce x en U, existe un programa de longitud $H(x) + i + 1$ que produce x en U^A. Por lo tanto podemos tomar $c = i + 1$. □

Proposición 2.19 *Todo número real r. e. es K-computable*

Demostración. Sea β un real r. e. El conjunto $B = \{p \in \mathbf{Q} \mid p < \beta\}$ es recursivamente enumerable, por lo tanto es K-computable. Entonces, para encontrar por ejemplo los primeros n bits de la expansión binaria de β simplemente hay que calcular $\max\{s| \ |s| = n \wedge \sum_{i=1}^{|s|} s_i 2^{-i} < \beta\}$, donde el máximo se toma con respecto al orden lexicográfico. Las cadenas de longitud n son un conjunto finito y B es K-computable, por lo que el conjunto sobre el que se busca el máximo es finito y K-computable. Por lo tanto, el máximo es K-computable. □

Capítulo 3

La probabilidad de programas válidos para cómputos infinitos

En este trabajo nos interesa estudiar la probabilidad de que una máquina M lea sólo una parte finita de la cinta de programa. Es decir, la probabilidad de que una secuencia elegida al azar comience con un programa válido para un cómputo infinito en M. Esta probabilidad es igual a:

$$\sum_{M^\infty(p)\downarrow} 2^{-|p|} = 2^{-K_M^\infty}$$

Al igual que con la probabilidad de detención, el caso interesante se da con las máquinas universales:

Definición 3.1 Llamamos α a la probabilidad de que una secuencia comience con un programa válido para un cómputo infinito en U. Es decir, $\alpha = 2^{-K^\infty}$.

Nuestro teorema central es que los bits de α son totalmente impredecibles, aun contando con un oráculo para el problema de la detención:

Teorema 3.2 *Existe una constante c tal que para todo n:*

$$H^K(\alpha(n)) > n - c$$

De lo que obtenemos también el siguiente corolario:

Corolario 3.3 *α es un número aleatorio no r. e.*

Demostración. Por el teorema 3.2 y la proposición 2.18, α es aleatorio. Y también por el teorema 3.2 α no puede ser K-computable, por lo que por la proposición 2.19 tampoco puede ser r. e. □

La estrategia de la demostración del teorema 3.2 será la siguiente: primero demostraremos que se puede establecer una correspondencia entre las secuencias que comienzan con un programa válido para U^K y un subconjunto de las secuencias que comienzan con un programa válido para cómputos infinitos en U. Luego aplicaremos ese resultado para mostrar que si se tienen los primeros n bits de α se puede resolver el problema de la detención para todos los programas de U^K de longitud menor o igual que n menos una constante. A partir de ahí se puede utilizar el mismo argumento que el utilizado en la prueba de aleatoriedad de Ω.

Teorema 3.4 *Existe un prefijo y tal que para toda secuencia $\boldsymbol{x}$, $U^K(\boldsymbol{x}) \downarrow$ si y solo si $U^\infty(y\boldsymbol{x}) \downarrow$.*

En la demostración utilizaremos la función $\mathrm{STP}_M : \Sigma^* \times \mathbf{N} \to \{0,1\}$ que definimos de la siguiente manera: $\mathrm{STP}_M(p,t)$ vale 1 si el cómputo para $M(p)$ se detiene exitosamente en t o menos ciclos de ejecución, y vale 0 en caso contrario. Esta función es recursiva total (en particular es recursiva primitiva). También utilizaremos una técnica llamada *simulación en el límite*, que fue introducida por Chaitin en [5]. Su aplicación para este resultado nos fue sugerida por él [8].

Demostración. y indica a U que realice la simulación en el límite del programa para U^K que viene a continuación. La simulación en el límite es un cómputo infinito y procede por pasos. En el paso t se efectúan los siguientes subpasos:

1. Se simulan t ciclos de ejecución de U^K. Durante la simulación, cada vez que se pregunta al oráculo si un programa q se detiene, se calcula $\mathrm{STP}_U(q,t)$ y se usa como respuesta.

2. Si en la simulación hecha en el subpaso 1 U^K no se detuvo, se avanza la cabeza de la cinta de programa hasta la posición t, si no se encontraba ya en esa posición. Si U^K se detuvo y lo hizo luego de n ciclos, se avanza la cabeza de la cinta de programa hasta la posición n, si no se encontraba ya en esa posición o más adelante.

Veamos que lo realizado por y cumple con lo pedido. Supongamos que $U^K(\mathbf{x}) \downarrow$. Esto quiere decir que hay un p que es prefijo de $\mathbf{x}$ tal que $U^K(p) \downarrow$. p debe detenerse en un tiempo finito. Por lo tanto, puede realizar una cantidad finita de consultas al oráculo. Llamemos Q al conjunto finito de programas sobre los cuales se consulta. Todo q en Q que cumpla $U(q) \downarrow$ debe detenerse en tiempo finito. Sea T_U la función parcial de dominio K y valores naturales tal que $T_U(q) = \mathrm{mín}\{t|\mathrm{STP}_U(q,t)\}$. Sea $m = \mathrm{máx}\{T_U(q)|q \in Q \cap K\}$. Si t es mayor o igual que m, la respuesta $\mathrm{STP}_U(q,t)$ coincidirá con la de $U(q) \downarrow$ para todo $q \in Q$. Entonces, a partir del paso m la simulación hecha en el subpaso 1 realizará lo mismo que U^K. Por lo que si $U^K(p)$ se detiene luego

de n pasos, cuando t supere al máximo entre m y n la simulación del paso t también se detendrá. Por lo tanto, la cabeza de la cinta de programa no avanzará más allá de la posición máx$\{m, n\}$. Entonces vale $U^\infty(y\mathbf{x})\downarrow$.

Supongamos ahora que $U^\infty(y\mathbf{x})\downarrow$. Esto quiere decir que durante la simulación la cabeza de la cinta de programa llega hasta cierta posición n y luego no avanza más. Queremos ver que $U^K(\mathbf{x})\downarrow$. Supongamos que no fuera así. Entonces la ejecución en U^K no se debe detener luego de $n+1$ ciclos. Durante ese lapso, se pueden haber efectuado solamente una cantidad finita de consultas al oráculo. Siguiendo el mismo razonamiento que antes, debe existir un tiempo m tal que $\mathrm{STP}_U(q, m)$ responde a todas esas consultas correctamente. Por lo tanto, a partir del tiempo máx$\{m, n+1\}$ la simulación debe hacer lo mismo que U^K en los primeros $n+1$ ciclos simulados, por lo que no se debe detener antes de $n+1$ ciclos y la cabeza de la cinta de programa será avanzada al menos hasta la posición $n+1$, lo que se contradice con la suposición original. Por lo tanto debe valer $U^K(\mathbf{x})\downarrow$. □

Corolario 3.5 *Sea y una cadena que cumpla con la propiedad enunciada en el teorema 3.4. Entonces vale:*

$$\Omega^K = \sum_{U^\infty(yp)\downarrow} 2^{-|p|}$$

Notación. Para cualquier cadena y y cualquier conjunto de cadenas X podemos separar a este último en dos conjuntos:

- $X_y = X \cap y\Sigma^*$, formado por las cadenas de X que comienzan con y.
- $X_{\bar{y}} = X \setminus y\Sigma^*$, formado por las cadenas de X que no comienzan con y.

Utilizando esta notación podemos reescribir el corolario 3.5 de la siguiente manera:

$$\Omega^K = 2^{-K^\infty_y} 2^{|y|} \tag{3.1}$$

Además, como $\{X_y, X_{\bar{y}}\}$ es una partición de X, se cumple que:

$$2^{-X} = 2^{-X_y} + 2^{-X_{\bar{y}}} \tag{3.2}$$

En la demostración del teorema 3.2 necesitaremos el hecho de que U^K es capaz de enumerar K^∞. Para demostrar esto ubicaremos a este conjunto dentro de la jerarquía aritmética (cf. Rogers [9, Cap. 14]), mostrando que está en Σ_2. (Un conjunto A está en Σ_2 si puede ser expresado mediante una fórmula de la forma: $A = \{z | \exists x_1 \ldots x_m \forall y_1 \ldots y_n R(z, x_1, \ldots, x_m, y_1, \ldots, y_n)\}$, siendo R un predicado recursivo.)

Lema 3.6 *Para toda máquina autodelimitante M: $K^\infty_M \in \Sigma_2$*

Demostración. Dada la descripción de una máquina, podemos definir la función $\mathrm{POS}_M : \Sigma^* \times \mathbf{N} \to \mathbf{N}$, tal que $\mathrm{POS}_M(p,t)$ devuelve la posición de la cabeza de la cinta de programa luego de t ciclos de ejecución de un cómputo para $M(p)$, si ésta no sobrepasa $|p|$, y devuelve $|p|+1$ en caso contrario.

Esta función es recursiva total (en particular es recursiva primitiva), y nos permite definir al conjunto K_M^∞ de la siguiente manera:

$$K_M^\infty = \{p | \exists t_0 \forall t_1 (t_1 \geq t_0 \to \mathrm{POS}_M(p,t_1) = |p|)\}$$

por lo que K_M^∞ está en Σ_2. □
Entonces, por el teorema de la jerarquía aritmética de Kleene-Post obtenemos:

Corolario 3.7 *Para toda máquina autodelimitante M, K_M^∞ es K-enumerable.*

Podemos demostrar ahora el resultado principal de este trabajo, teorema 3.2:
Demostración. Supongamos que y cumple con la propiedad enunciada en el teorema 3.4. Consideremos un programa para U^K formado por un prefijo w y un programa de longitud mínima para $\alpha(n)$, en el que w hace que U haga lo siguiente:

1. Lee y ejecuta el programa que viene a continuación, obteniendo $\alpha(n)$.
2. Comienza una enumeración del conjunto K^∞. Escribiremos $K^\infty[t]$ para referirnos al conjunto formado por los primeros t elementos enumerados. La enumeración se detiene al encontrar un t tal que $2^{-K^\infty[t]} > \alpha(n)$.
3. Obtiene $K_y^\infty[t] = K^\infty[t] \cap y\Sigma^*$.
4. Calcula $\beta = 2^{-K_y^\infty[t]} 2^{|y|}$.
5. Comienza una enumeración del conjunto K'. Llamaremos $K'[r]$ al conjunto formado por los primeros r elementos enumerados. La enumeración se detiene al encontrar un r tal que $2^{-K'[r]} > \beta$.
6. Imprime como resultado la menor cadena z que no se encuentre en el conjunto $S = \{s | U^K(p) = s \wedge p \in K'[r]\}$, y termina.

Queremos ver que $H^K(z) > n - |y|$. Supongamos que no fuera así. Entonces debería haber un programa p para U^K tal que $U^K(p) = z$ y $|p| \leq n - |y|$. Como $z \notin S$, p no puede pertenecer a $K'[r]$. Pero al mismo tiempo $p \in K'$. Por lo tanto,

$$2^{-K'} - 2^{-K'[r]} \geq 2^{-|p|} \geq 2^{-n+|y|}$$

Pero por el paso 5 teníamos que $2^{-K'[r]} > 2^{-K_y^\infty[t]} 2^{|y|}$, por lo que nos queda:

$$2^{-K'} - 2^{-K_y^\infty[t]} 2^{|y|} > 2^{-n+|y|}$$

Además, por la ecuación (3.1), $2^{-K'} = 2^{-K_y^\infty} 2^{|y|}$. Entonces:

$$2^{-K_y^\infty} 2^{|y|} - 2^{-K_y^\infty[t]} 2^{|y|} > 2^{-n+|y|}$$

Dividiendo por $2^{|y|}$:

$$2^{-K_y^\infty} - 2^{-K_y^\infty[t]} > 2^{-n} \tag{3.3}$$

Por otra parte, por la ecuación (3.2):

$$2^{-K^\infty} - 2^{-K^\infty[t]} = \left(2^{-K_y^\infty} + 2^{-K_{\tilde{y}}^\infty}\right) - \left(2^{-K_y^\infty[t]} + 2^{-K_{\tilde{y}}^\infty[t]}\right)$$

Reagrupando:

$$2^{-K^\infty} - 2^{-K^\infty[t]} = \left(2^{-K_y^\infty} - 2^{-K_y^\infty[t]}\right) + \left(2^{-K_{\tilde{y}}^\infty} - 2^{-K_{\tilde{y}}^\infty[t]}\right)$$

Como $K_{\tilde{y}}^\infty \supset K_{\tilde{y}}^\infty[t]$, el segundo término de la expresión de la derecha es mayor que 0, por lo que obtenemos:

$$2^{-K^\infty} - 2^{-K^\infty[t]} > 2^{-K_y^\infty} - 2^{-K_y^\infty[t]} \tag{3.4}$$

Combinando (3.3) y (3.4):

$$2^{-K^\infty} - 2^{-K^\infty[t]} > 2^{-n}$$

Pero $2^{-K^\infty} = \alpha$, y habíamos elegido a t de manera que $2^{-K^\infty[t]} > \alpha(n)$, por lo que resulta:

$$\alpha - \alpha(n) > 2^{-n}$$

Lo cual es absurdo, ya que $\alpha(n)$ está formado por los primeros n dígitos de α, por lo que la diferencia entre ambos debe ser menor que 2^{-n}. El absurdo provino de suponer que existía un programa p para U^K tal que $U^K(p) = z$ y $|p| \leq n - |y|$. Por lo tanto debe valer:

$$H^K(z) > n - |y|$$

y como z se obtiene de un programa formado por un prefijo w y un programa mínimo para $\alpha(n)$:

$$H^K(z) \leq |w| + H^K(\alpha(n))$$

Por lo que obtenemos:

$$H^K(\alpha(n)) > n - |y| - |w|$$

Entonces, tomando $c = |y| - |w|$, obtenemos lo que queríamos demostrar. □

Capítulo 4

Conclusiones y trabajo futuro

En este trabajo hemos demostrado que la probabilidad de que una secuencia comience con un programa válido para un cómputo infinito en una máquina universal de Chaitin es un número totalmente impredecible, incluso contando con un oráculo para el problema de la detención. La clase de números reales que tienen esta propiedad —a los que podríamos llamar K-aleatorios— aún no ha sido convenientemente explorada. Si bien Ω^K es un ejemplo inmediato, ya que la prueba de aleatoriedad de Ω se puede relativizar sin inconvenientes para mostrar que Ω^K es K-aleatorio, este es el primer ejemplo que conocemos que surge de una definición distinta. No es trivial encontrar ejemplos con estas características, ya que una relativización de la proposición 2.15 nos muestra que no cualquier conjunto A de Σ_2 tal que K' sea A-recursivo tiene probabilidad K-aleatoria. Como trabajo inmediato, nos proponemos analizar las probabilidades de otros conjuntos que son equivalentes a K' y K^∞ desde el punto de vista de la teoría de la recursión, como la probabilidad de que una máquina enumere un conjunto finito, o la probabilidad de que una máquina compute una función parcial. Creemos que estos números tienen las mismas propiedades de aleatoriedad que α, por lo que podrían ser vistos como definiciones alternativas. Otra tarea sería ver si se pueden usar técnicas similares para probabilidades de conjuntos que se encuentran más arriba en la jerarquía aritmética, como por ejemplo la probabilidad de que una máquina enumere un conjunto co-finito.

También sería interesante extender los resultados obtenidos en [2] y [13] para obtener una versión relativizada del teorema 2.16 que diga:

> Si β es un número A-r. e. y cumple $H^A(\beta(n)) > n - c$ entonces β es la probabilidad de detención de una máquina universal de Chaitin con oráculo para A.

Y entonces concluir que α es la probabilidad de detención de alguna máquina universal de Chaitin con oráculo para K.

Otra tarea a realizar sería continuar desarrollando la teoría de complejidad de conjuntos iniciada por Chaitin en [5], a la que Solovay hizo algunos aportes en [14]. En esta teoría se mide la complejidad de un conjunto por la longitud del programa para cómputos infinitos más corto que lo genere. En ese contexto se puede ver a nuestro trabajo como el análogo en cómputos infinitos del teorema de aleatoriedad de la probabilidad de detención para cómputos finitos.

Agradecimientos

Quiero agradecer a Gregory Chaitin por haberme introducido a la Teoría Algorítmica de la Información, y por su constante apoyo y sus cruciales sugerencias durante la realización de este trabajo.

A Verónica Becher, por haber planteado el problema a investigar en su forma inicial y por estar siempre dispuesta a discutir ideas, aportar sugerencias, y a leer y releer las distintas versiones de este trabajo.

A los profesores Roberto Cignoli, Jorge Aguirre, Argimiro Arratia, Gregory Chaitin, Marcelo Scasso, Xavier Caicedo y Cristian Calude, por mi formación en lógica y teoría de la computación, en distintos momentos de mi carrera.

A Carlos Diuk, María Elinger, Carlos López Holtman y Sergio Zlotnik, por hacer llevaderas las incontables horas preparando trabajos prácticos a lo largo de la carrera.

A Diego Pol, por ayudarme a conseguir algunos de los papers citados en la bibliografía.

A Diego Garbervetsky, por su apoyo durante la finalización de este trabajo, y por sus sugerencias para mejorar la claridad del mismo.

Bibliografía

[1] Cristian. S. Calude. *Information and Randomness. An Algorithmic Perspective*. Springer-Verlag, Berlin, 1994.

[2] Cristian S. Calude, Peter H. Hertlind, Bakhadyr Khoussainov, and Yongee Wang. Recursively enumerable reals and Chaitin Ω numbers. *Theoret. Comput. Sci.* In press.

[3] G. J. Chaitin. On the length of programs for computing finite binary sequences. *J. Assoc. Comput. Mach.*, 13:547–569, 1966.

[4] G. J. Chaitin. A theory of program size formally identical to information theory. *J. Assoc. Comput. Mach.*, 22:329–340, 1975.

[5] G. J. Chaitin. Algorithmic entropy of sets. *Computers & Mathematics with Applications*, 2:233–245, 1976.

[6] G. J. Chaitin. Algorithmic information theory. *IBM J. Res. Develop.*, 21(496):350–359, 1977.

[7] G. J. Chaitin. *Algorithmic Information Theory*. Cambridge University Press, Cambridge, England, 1987.

[8] G. J. Chaitin. Comunicación personal, Febrero 2000.

[9] Hartley Rogers Jr. *Theory of Recursive Functions and Effective Computability*. MIT Press, Cambridge, Massachusetts, first paperback edition, 1987.

[10] A. N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Inform. Transmission*, 1(1):1–7, January 1965.

[11] L. G. Kraft. A device for quantizing, grouping and coding amplitude modulated pulses. Master's thesis, Dept. of Electrical Engineering, M.I.T., Cambridge, Massachusets, 1949.

[12] P. Martin-Löf. The definition of random sequences. *Information & Control*, 9:602–619, 1966.

[13] Theodore A. Slaman. Randomness and recursive enumerability. Preprint, 1999.

[14] R. M. Solovay. On random r. e. sets. In *Proceedings of the Third Latin American Symposium on Mathematical Logic*, Campinas, Brazil, July 1976.