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

Martingalas Computables y Secuencias Genéricas de Poisson

Tesis de Licenciatura en Ciencias de la Computación

Franco Assenza

Directores: Verónica Becher y Santiago Figueira
Buenos Aires, 12 de marzo de 2024

MARTINGALAS COMPUTABLES Y SECUENCIAS GENÉRICAS DE POISSON

Intuitivamente un número es aleatorio si no hay forma hacer apuestas a los dígitos de su expansión fraccionaria y conseguir, a la larga, una ganancia infinita. La formalización de esta idea originó la definición de aleatoriedad basada en martingalas. Un número es puramente aleatorio si ninguna martingala computablemente enumerable hace una ganancia infinita sobre su expansión fraccionaria. Las nociones de aleatoriedad más débiles (o impuras) sí pueden tener martingalas computables asociadas. En este trabajo consideramos la noción llamada genericidad de Poisson. Un número es genérico de Poisson si la distribución de bloques largos de dígitos en los segmentos iniciales de su expansión fraccionaria es una distribución de Poisson. Recientemente Peres y Weiss demostraron que casi todos los números son genéricos de Poisson, y Álvarez, Becher y Mereb mostraron que todos los números puramente aleatorios son genéricos de Poisson. En esta tesis damos una martingala computable que logra infinita ganancia sobre las expansiones fraccionarias de números que no son Poisson genéricos en una base entera dada, analizamos su complejidad computacional de peor caso. Esto prueba que alcanza 2^{n2^n}-aleatoriedad para garantizar genericidad de Poisson.

Palabras clave: aleatoriedad algorítmica; distribución de Poisson; test de Martin-Löf

AGRADECIMIENTOS

La cantidad de gente que estuvo a mi lado y que me ayudó a aprender a pensar es enorme. La costumbre dicta aprovechar este pequeño espacio para agradecer a muchos de ellos, que son gente muy importante para mí y que quiero un montón. Me da terror tener que armar una lista porque soy yo, y es obvio que va a faltar gente importante, incluso limitándome a lo que tengo alguna excusa de relacionarlos a la academia. Igual, si estás leyendo esto, hay probabilidad alta de que tendrías que estar en esta lista.

A mamá, papá, Mica, Tomy. A diversas distancias físicas, siempre apoyándome. Es un montón lo que aprendí de ellos, y son quienes me inspiraron el amor por la matemática y la computación.

A la inmensa cantidad de gente de la hermosa comunidad de olimpíadas que, aunque los años indican que fue algo que pasó hace unos cuántos años, lo sigo sintiendo cercano. Los primeros nombres que vienen a la mente para agradecer por todo el esfuerzo y amor que le ponen a la comunidad son los de Patricia y Flora, pero la lista es inmensa, abarcando gente que ayuda a que la comunidad exista y olímpicos con los que compartí algún viaje o problema. Vero, Marita, Néstor, Mati, Ari, Caro, Mel, Nico, Margy, Maite, Ale, Gasti, Seba, Miguel, Freddy, Rocío, Lupi, Dani, Iara, Agus, Lauti, Nati, Iván, Mariano, y mucha, mucha gente más.

A mis compañeros de BRS. Especialmente tengo que nombrar a Lau, Lean y a Luqui, que me recibieron con mucho cariño en lo que fue un cambio enorme, y con quienes trabajamos juntos habiendo empezado hace ya más de 12 años.

A quien hoy es la persona más importante, mi compañera y empuje en este y muchos otros aspectos, Aye. A mis suegros, Danilo y Silvia, y a Nahuel, que me recibieron y siguen recibiendo hace años.

A mis compañeros de docencia en Algo 3 con quienes compartí estos últimos años, especialmente los más longevos en la materia: Fran, Pablo, Santi, Juli, Tropi, Tomás, Freddy. Y a los chicos que me tuvieron de docente, que aprendí un montón de contarles cosas, especialmente de a uno, y ojalá haya quedado una sensación similar.

A quienes me ayudaron en este último paso. Vero, Santi, fue muy divertido pensar problemitas con ustedes. Martín, Nico, gracias por sumarse, haber revisado y por sus comentarios. Y a la facu en general, con todo lo que eso abarca y significa. Ojalá pueda seguir siendo lo que es para tanta gente.

Índice general

1. INTRODUCCIÓN

Intuitivamente, un número real es aleatorio cuando tiene todas las propiedades que se cumplen para casi todos los números reales. Por ejemplo, su expansión decimal no tiene regularidades predecibles. Entonces, un número real aleatorio debe pasar todos los tests de estas propiedades que se cumplen para casi todo número real. La formalización de esta idea resulta en la definición de aleatoriedad basada en tests computablemente enumerables dada por Martin Löf en 1965, ver [5].

Otra intuición nos dice que un número real es aleatorio si la única forma de describir su expansión decimal es explícitamente. La formalización de esta intuición resultó en la definición de aleatoriedad basada en la complejidad de largo de programa dada por Gregory Chaitin en 1975, ver [5].

Aún otra intuición nos dice que un número real es aleatorio si no hay forma de hacer apuestas a los dígitos de su expansión fraccionaria y hacer, a la larga, una ganancia infinita. La formalización de esta idea originó la definición de aleatoriedad basada en martingalas computablemente enumerables anticipada por Jean Ville en los años 30 y formalizada por Claus Peter Schnorr en 1975. La historia de esta formalización fue publicada recientemente en [4].

Dado que las tres formalizaciones de aleatoriedad resultaron ser equivalentes[7], se considera que tales formalizaciones dan una definición matemática de numero real aleatorio. Se considera que esta definición es robusta porque estas tres formalizaciones del concepto son equivalentes pero muy distintas entre sí.

Existen definiciones de aleatoriedad más restrictivas, más específicas. Si pedimos que cualquier secuencia de dígitos aparezca con frecuencia límite equivalente a otras del mismo tamaño, estamos abarcando la definición de normalidad, que también trae consigo definiciones equivalentes en el mundo de los compresores y las apuestas. Si pedimos computabilidad a la definición de Martin-Löf, estamos definiendo aleatoriedad de Schnorr. Si pedimos que estrategias computables de apuestas no puedan ganar, estamos hablando de aleatoriedad computable. Si pedimos que las estrategias de apuestas sean computables en $O(t(n))$ estamos hablando de $t(n)$-aleatoriedad

Zeev Rudnick define un tipo de aleatoriedad intermedia, que llama supernormalidad, como aquellos números en los que la distribución de bloques largos de dígitos en segmentos iniciales de sus expansiones fraccionarias siguen una distribución de Poisson. En 2022, Yuval Peres y Benjamin Weiss demostraron que casi todos los números reales son supernormales, ahora llamados Poisson genéricos. En 2023, Nicolás Álvarez, Verónica Becher y Martín Mereb dieron un test de Martin-Löf que contiene a todos los números de que no son genéricos de Poisson.

En esta tesis exhibimos una martingala computable que tiene éxito en los números que no son Poisson genéricos en una base entera dada, y analizamos su complejidad computacional de peor caso.

2. NOTACIÓN Y DEFINICIONES BÁSICAS

2.1. Notación

Suponemos un alfabeto finito Σ, por ejemplo el alfabeto de dos símbolos. El conjunto de strings finitos es Σ^*, el conjunto de los strings infinitos es $\Sigma^{\mathbb{N}}$. Usamos las letras $\sigma, \rho, \tau, \omega, x, y, z$ para referirnos a strings finitos o infinitos. Contamos las posiciones de un string comenzando en 1. Si σ es un string finito entonces $|\sigma|$ es su longitud. Si $\sigma \in \Sigma^*, \tau \in \Sigma^* \cup \Sigma^{\mathbb{N}}$, $\sigma\tau$ es la concatenación de strings σ y τ, y si $x \in \Sigma$, σx es la concatenación del string σ con el símbolo x.

Escribimos $x \upharpoonright_n$ para el string de los primeros n caracteres del string x. De manera similar, $x[i, \ldots, j]$ refiere a los caracteres entre la posición i y la j. $|x|_s$ denota la cantidad de veces que aparece el string s dentro del string x.

Usamos el símbolo $\preceq$ para denotar la relación de ser prefijo. Es decir, $\sigma \preceq \tau$ si y solo si $\exists \rho, \sigma\rho = \tau$. En el caso en el que $\sigma \in \Sigma^*, \tau \in \Sigma^{\mathbb{N}}$ notaremos $\sigma \prec \tau$

Llamaremos b a la cardinalidad del alfabeto Σ. Sin pérdida de generalidad si el alfabeto tiene b símbolos, los llamaremos $0, 1, \ldots, b-1$. Cuando Σ es el alfabeto binario, $\Sigma^{\mathbb{N}}$ es el espacio de Cantor usualmente notado 2^ω.

Un **racional diádico** en base r es un racional con representación finita en base r, es decir de la forma $z \cdot r^{-n}$ para algún $z \in \mathbb{Z}, n \in \mathbb{N}$. Sea $\mathrm{Rat}_r^{\geq 0}$ el conjunto de los racionales diádicos no-negativos en base r. En este trabajo usaremos $r = 2$.

En general vamos a estar trabajando en base $b = 2$, aclarando base b donde corresponda.

El material de este capítulo se puede leer de [1] y [7].

2.2. Espacio de Cantor y conjuntos abiertos

Un conjunto abierto básico de $\Sigma^{\mathbb{N}}$ es un conjunto $[y] = \{Z \in \Sigma^{\mathbb{N}} : y \prec Z\}$, con $y \in \Sigma^*$.

El conjunto abierto generado por un conjunto $S \subseteq \Sigma^\star$ es

$$[S] = \{X \in \Sigma^{\mathbb{N}} : \exists y \in S, y \prec X\}$$

2.3. Medidas

Llamamos *medida* a una función $r : \Sigma^\star \to \mathbb{R}_0^+$ que satisfaga para todo $\sigma \in \Sigma^\star$ la igualdad

$$\sum_{x \in \Sigma} r(\sigma x) = r(\sigma).$$

Una *medida externa* es una función $\mu : \mathbb{P}(\Sigma^{\mathbb{N}}) \to \mathbb{R}_0^+$ que satisface

- $\mu(\emptyset) = 0$

- $C \subseteq D \subseteq \Sigma^{\mathbb{N}} \to \mu(C) \leq \mu(D)$ (monotonicidad)
- $\mu(\cup_i C_i) \leq \sum_i \mu(C_i)$ para familias $(C_i)_{i\in\mathbb{N}}$.

Podemos extender una medida r a una medida externa μ_r haciendo lo siguiente,

1. Si $A \subseteq \Sigma^{\mathbb{N}}$ es un abierto, elijamos un conjunto libre de prefijos $E \subseteq \Sigma^\star$ tal que $[E] = A$. En ese caso, sea $\mu_r(A) = \sum_{\sigma\in E} r(\sigma)$
2. Para cualquier $C \subseteq \Sigma^{\mathbb{N}}$ sea

$$\mu_r(C) = \text{ínf}\{\mu_r(A) : C \subseteq A \ \&\ A \text{ es abierto } \}$$

A la extensión de la medida $r(x) = b^{-|x|}$ se la suele denotar λ y llamar medida uniforme externa. Para evitar colisión con otras λ de este texto, aquí usaremos μ.

Definición 1. *Un conjunto $G \in \Sigma^{\mathbb{N}}$ es medible respecto a una medida μ si para todo $C \subseteq \Sigma^{\mathbb{N}}$ sucede que $\mu(C) = \mu(C \cap G) + \mu(C - G)$.*

Definición 2. *La medida condicional entre 2 abiertos A y B se define como*

$$\mu(A|B) = \tfrac{\mu(A\cap B)}{\mu(B)}.$$

2.4. Computabilidad y tests

Definición 3. *Sea ψ una función que tenga como dominio un subconjunto de $\mathbb{N}^k$ y como imagen un subconjunto de $\mathbb{N}$. Decimos que ψ es **parcialmente computable** si existe una máquina de Turing P con k cintas tal que $\psi(x_0,\ldots,x_{k-1}) = y$ si y solo si P en inputs $x_0,\ldots,x_{k-1}$ retorna y. Notamos $\psi(x_0,\ldots,x_{k-1}) \downarrow$ si P termina con inputs $x_0,\ldots,x_{k-1}$. Decimos que ψ es **computable** si ψ es parcialmente computable y su dominio es $\mathbb{N}^k$.*

Junto con la tesis de Church–Turing, procedimientos algorítmicos informalmente descriptos pueden ser implementados por una máquina de Turing. Esto significa que para probar computabilidad de una f nos alcanza con dar un procedimiento algorítmico que compute f.

Definición 4. *Un conjunto $A \subseteq \mathbb{N}$ es **computablemente enumerable** (abreviado c.e.) si es el dominio de alguna función parcialmente computable. Una secuencia de conjuntos $(S_e)_{e\in\mathbb{N}}$ tal que el conjunto de pares $\{\langle e, x\rangle : x \in S_e\}$ es c.e. es llamado **uniformemente computablemente enumerable**.*

Definición 5.
1. *Un **test de Martin-Löf** es una secuencia uniformemente computablemente enumerable de abiertos $(G_m)_{m\in\mathbb{N}}$ tal que $\forall m \in \mathbb{N}, \mu G_m \leq 2^{-m}$*
2. *Un string $Z \in \Sigma^{\mathbb{N}}$ se dice que **falla el test** si $Z \in \cap_m G_m$. En otro caso se dice que el string **pasa el test**. Un conjunto de strings falla el test si alguno de sus strings falla el test. En caso contrario, pasa el test.*

Definición 6. *Llamamos a un string infinito $x \in \Sigma^{\mathbb{N}}$ **Martin-Löf aleatorio** si pasa todo test de Martin-Löf.*

Definición 7. *Un **test de Schnorr** es un Test de Martin-Lof $(G_m)_{m\in\mathbb{N}}$ tal que μG_m es computable uniformemente en m.*

2.5. Martingalas

Una **martingala** es una función $B : \Sigma^\star \to \mathbb{R}_{\geq 0}$ que satisface para todo $x \in \Sigma^\star$

$$\sum_{y \in \Sigma} B(xy) = bB(x)$$

donde b es la cardinalidad del alfabeto Σ.

Esta definición formaliza el concepto de apuesta. Por ejemplo, en el caso de 2 símbolos, tendríamos un capital inicial $B(x)$ y "apostamos" $B(x0) - B(x)$ a un próximo elemento 0. Esta definición es una extensión de la descripta en [7] con 2 símbolos.

Una martingala **tiene éxito** en un string infinito $Z \in \Sigma^{\mathbb{N}}$ si $\sup_n B(Z \upharpoonright_n)$ no está acotado.

Como no podemos computar números reales directamente, hacemos aproximaciones

Definición 8. *Llamemos $\hat{B} : \Sigma^* \times \mathbb{N} \to Rat_2^{\geq 0}$ a una **función de aproximación** de $B : \Sigma^* \to \mathbb{R}_{\geq 0}$, con precisión 2^{-n}.*

Es decir, $\hat{B}$ es una función que cumple que para todo $x \in \Sigma^*, n \in \mathbb{N}$, sucede que

$$|\hat{B}(x, n) - B(x)| < 2^{-n}$$

.

*La complejidad de $\hat{B}$ en (x, n) se mide en función de $|x| + |n|$. Una $t(n)$-**aproximación computable** es una aproximación computable determinística en tiempo $O(t(n))$.*

Llamamos a B una $t(n)$-**martingala** si B es una martingala computable en tiempo $O(t(n))$ o, equivalentemente, si existe una aproximación computable en tiempo $O(t(n))$.

Definición 9. *Un string infinito es $t(n)$-**aleatorio** si ninguna $t(n)$-martingala tiene éxito en dicho string.*

Por [6], son equivalentes las nociones de martingala con una $t(n)$-aproximación computable y una martingala $t(n)$-computable, si existe una existe la otra.

Observación 1. *Sea $B : \Sigma^\star \to \mathbb{R}_{\geq 0}$ y sea $r : \Sigma^\star \to \mathbb{R}_{\geq 0}$ una función $r(x) = b^{-|x|}B(x)$, entonces B es martingala si y solo si r es medida.*

Demostración. Se ve de las definiciones, ya que, con $x \in \Sigma^\star$, $n = |x|$,

$$\sum_{y \in \Sigma} b^{-(n+1)} B(xy) = b^{-n} B(x) \text{ si y solo si } \sum_{y \in \Sigma} r(xy) = r(x).$$

□

Entonces,

Observación 2. *Cualquier conjunto medible $C \subseteq \Sigma^{\mathbb{N}}$ define una martingala asociada a su medida.*

Demostración. La medida en C, $\mu_C(A) = \mu(C \cap A)$ para A medible, tiene como representación $r(x) = \mu(C \cap [x])$ y entonces definimos

$$B_C(x) = b^{|x|} r(x)$$

que es una martingala, y también una probabilidad condicional, y representa la probabilidad de caer en C partiendo de x. Además, las martingalas dan una caracterización de los conjuntos de medida 0. □

Observación 3. *El conjunto $A \subseteq \Sigma^{\mathbb{N}}$ tiene medida 0 si y solo si existe sucesión de conjuntos abiertos $(G_i)_{i\in\mathbb{N}}$ tal que $\mu G_i \leq 2^i$ y $A \subseteq \cap G_i$.*

Lema 1. *$A \in \Sigma^{\mathbb{N}}$ es Martin-Löf aleatorio si y solo si no existe una martingala computablemente enumerable B que tiene éxito en A.*

Observación 4 ([7, Proposición 7.2.6])**.** *Si existe un Test de Martin-Lof $(G_i)_{i\in\mathbb{N}}$ donde los G_i tienen medida computable, la martingala $B = \sum B_{G_i}$ tiene éxito en A y es computable.*

Observación 5. *Si $b = 2$, para todo $x \in \Sigma^*, n \in \mathbb{N}$ sucede que* $\sum_{i>|x|+n} B_{G_i}(x) \leq 2^{-n}$

Demostración. $B_{G_i}(x) = b^{|x|}\mu(G_i \cap [x]) \leq b^{-i+|x|} \leq 2^{-i+|x|}$, con lo que

$$\sum_{i>|x|+n} B_{G_i}(x) \leq 2^{-n}.$$

□

Notar que para b arbitrario podemos hacer una demostración similar, que requerirá pedir $i > n + |x| \log_2(b)$ en vez de $i > n + |x|$.

Observación 6. *Si $b = 2$, una posible función que aproxima B es*

$$\hat{B}(x, n) = \sum_{i\leq|x|+n+1} \hat{B}_{G_i}(x, n + i + 2).$$

Demostración. Dado que

$$\left|\hat{B}_{G_i}(x, n + i + 2) - B_{G_i}(x)\right| < 2^{-(n+i+2)},$$

tenemos que

$$\left|\hat{B}(x, n) - \sum_{i\leq|x|+n+1} B_{G_i}(x)\right| < 2^{-(n+1)}.$$

Luego,

$$\begin{aligned}
\left|\hat{B}(x,n) - B(x)\right| &= \left|\hat{B}(x,n) - B(x) + \sum_{i\leq|x|+n+1} B_{G_i}(x) - \sum_{i\leq|x|+n+1} B_{G_i}(x)\right| \\
&= \left|\hat{B}(x,n) + \sum_{i>|x|+n+1} B_{G_i}(x) - \sum_{i\leq|x|+n+1} B_{G_i}(x)\right| \\
&\leq \left|\hat{B}(x,n) - \sum_{i\leq|x|+n+1} B_{G_i}(x)\right| + \left|\sum_{i>|x|+n+1} B_{G_i}(x)\right| \\
&< 2^{-(n+1)} + 2^{-(n+1)} \\
&< 2^{-n}.
\end{aligned}$$

□

El caso de b arbitrario, similarmente, requiere computar considerando hasta

$$i \leq n\log_2(b) + 1.$$

2.6. Ley de Poisson

Definición 10. *La **distribución de Poisson** con parámetro $\lambda > 0$, que denotamos Po_λ, es una distribución de probabilidades sobre una variable discreta X tal que para cada $j = 0, 1, 2, ...$, la probabilidad de que X sea igual a j es*

$$Po_\lambda(j) = e^{-\lambda}\lambda^j/j!.$$

2.7. De números reales a strings

Dado un número real x y una base entera b mayor o igual que 2, la expansión fraccionaria de x es la secuencia $a_1, a_2, \ldots$ donde cada $a_j \in \{0, .., b-1\}$ y

$$x = \lfloor x \rfloor + \sum_{j=1} a_j b^{-j}.$$

Dado que trabajamos con las expansiones fraccionarias de los números en una base entera dada, daremos la definición de Poisson genérico para strings infinitos sobre un alfabeto dado, que generalmente será $b = 2$.

2.8. Strings de Poisson

Sea Σ un alfabeto de b símbolos. Un string infinito $x \in \Sigma^{\mathbb{N}}$ es **Poisson genérico** si para cada real positivo λ y para cada número entero $j \geq 0$,

$$\lim_{k\to\infty} \frac{\#\{s \in \Sigma^* : |s| = k, |x\restriction_{\lfloor \lambda b^k \rfloor + k}|_s = j\}}{b^k} = Po_\lambda(j). \tag{2.1}$$

Vamos a usar $I_j(x,\omega)$ como la indicadora de que el string $\omega \in \Sigma^*$ aparece en x en la posición j,

$$I_j(x,\omega) = \mathbb{1}_{\{x[j,j+|\omega|)=\omega\}}.$$

Escribimos $Z^\lambda_{j,k}(x)$ para la proporción de strings de longitud k que aparecen j veces en el segmento inicial de la string x de longitud $\lfloor \lambda b^k \rfloor + k$. Es decir, con $|x| = \lfloor \lambda b^k \rfloor + k$,

$$Z^\lambda_{i,k}(x) = \frac{1}{b^k} \left\{ w \in \Sigma^k : \sum_{1 \le j \le \lambda b^k} I_j(x,\omega) = i \right\}.$$

Notar que la expresión (2.1) es equivalente a $\lim_{k\to\infty} Z^\lambda_{j,k}(x) = Po_\lambda(j)$.

En [2] hay una construcción sencilla y computable de un string infinito λ-Poisson genérico, para cualquier alfabeto Σ y cualquier $\lambda > 0$ positivo, excepto para el alfabeto binario que se requiere $\lambda \le \ln 2$.

Definición 11. *Dados parámetros $\lambda \in \mathbb{R}, k \in \mathbb{N}, j \in \mathbb{N}_0, \epsilon \in \mathbb{Q}_{[0,1]}$ definimos como $Bad(\lambda,k,j,\epsilon)$ al conjunto de strings de longitud $\lfloor \lambda b^k \rfloor + k$ que se alejan de la ley de Poisson para j ocurrencias más que ϵ:*

$$Bad(\lambda,k,j,\epsilon) = \left\{ \omega \in \Sigma^{\lfloor \lambda b^k \rfloor + k} : \left| Z^\lambda_{j,k}(\omega) - Po_\lambda(j) \right| > \epsilon \right\}.$$

Definición 12. *Definimos $Bad_k \subseteq \Sigma^*$ para cada k, que reúne una cantidad finita de conjuntos $Bad(\lambda,k,j,\epsilon)$, variando los otros parámetros, λ, j y ϵ.*

$$\begin{aligned} Bad_k &:= \bigcup_{j \in J_k} \bigcup_{\lambda \in L_k} Bad(\lambda,k,j,2/k) \\ J_k &:= \{0,\dots,b^k-1\} \\ L_k &:= \{p/q : q \in \{1,\dots,k\}, p/q < k\}. \end{aligned}$$

Notación. *Abusaremos la notación y diremos que un string $\omega \in Bad_k$ cuando $\omega \restriction_{\lfloor \lambda b^k \rfloor + k} \in Bad_k$.*

Definición 13. *Definimos $Bad_{k,\lambda} \in \Sigma^*$ para cada k como la versión fijando λ en Bad_k. Es decir*

$$\begin{aligned} Bad_{k,\lambda} &:= \bigcup_{j \in J_k} Bad(\lambda,k,j,2/k) \\ J_k &:= \{0,\dots,b^k-1\} \end{aligned}$$

3. UNA MARTINGALA COMPUTABLE

Este capítulo está centrado en base $b = 2$, con la versión de base b indicada donde corresponda.

La idea afortunada va a ser sumar muchas martingalas que pierden poco y ganan mucho.

Teorema 1. *2^{n2^n}-aleatoriedad implica Poisson genérico.*

3.1. De aleatoriedad a existencia de martingalas

Esta implicación va a estar dada por su contrarrecíproco, es decir, vamos a demostrar que

Teorema 2. *Existe una 2^{n2^n}-martingala B que tiene éxito en los strings infinitos que no son Poisson genéricos.*

3.2. ¿Cuál es la martingala?

La martingala en cuestión es la suma de martingalas más pequeñas $B(\omega) = \sum B_k(\omega)$, donde B_k apuesta a que el string pertenece al conjunto descripto en Bad_k. Formalmente,

Para cada $k \in \mathbb{N}$ definimos $B_k : \Sigma^* \to \mathbb{R}_{\geq 0}$,

$$B_k(\omega) = 2^{|\omega|}\mu(Bad_k \cap [\omega]) = \mu(Bad_k|\omega).$$

También definimos, para cada $k \in \mathbb{N}, \lambda \in \mathbb{R}^+$, $B_{k,\lambda} : \Sigma^* \to \mathbb{R}_{\geq 0}$,

$$B_{k,\lambda}(\omega) = 2^{|\omega|}\mu(Bad_{k,\omega} \cap [\omega]) = \mu(Bad_{k,\lambda}|\omega)$$

3.3. B es una martingala

Para esto, al ser suma de martingalas, alcanza con ver que comienza con capital finito.

Lema 2. $B(\emptyset) < \infty$.

Demostración. El trabajo [1] introduce los conjuntos Bad_k usando $1/k$ y se demuestra una cota superior de su medida. Dado que

$$\left\{\omega \in \Sigma^{\lfloor \lambda b^k \rfloor + k} : \left| Z^{\lambda}_{j,k}(\omega) - Po_\lambda(j) \right| > \frac{2}{k} \right\} \subseteq \left\{\omega \in \Sigma^{\lfloor \lambda b^k \rfloor + k} : \left| Z^{\lambda}_{j,k}(\omega) - Po_\lambda(j) \right| > \frac{1}{k} \right\}$$

usamos aquí $2/k$ en vez de $1/k$. Usaremos la función

$$N_n = b^{2n}.$$

En [1] se demuestra que existe un n_0 tal que para todo $n \geq n_0$,

$$\mu(Bad_{N_n}) < \frac{1}{2N_n^2}$$

y que

$$\mu\left(\bigcup_{N_n \leq k < N_{n+1}} Bad_k\right) < 2\mu(Bad_{N_n}).$$

Llamemos $f(n) = \bigcup_{N_n \leq k < N_{n+1}} Bad_k$. Luego

$$\begin{aligned}
B(\emptyset) &= \sum \mu(Bad_k) \\
&= \sum f(n) \\
&= \sum_{n<n_0} f(n) + \sum_{n \geq n_0} f(n) \\
&< \sum_{n<n_0} f(n) + \sum_{n \geq n_0} 2\mu(Bad_{N_n}) \\
&< \sum_{n<n_0} f(n) + \sum_{n \geq n_0} \frac{1}{N_n^2} \\
&< \sum_{n<n_0} f(n) + \sum_{n \geq n_0} \frac{1}{n^2} < \infty.
\end{aligned}$$

□

3.4. B tiene éxito en las secuencias que no son Poisson genéricas

Observación 7. *Si ω no es Poisson-genérica entonces existe una secuencia creciente $(k_i)_{i \in \mathbb{N}}$ tal que $\omega \in Bad_{k_i}$ para todo i.*

Demostración. Si el string no es Poisson-genérico, el límite de la expresión (2.1) no se cumple y han de existir λ, j y un $\epsilon > 0$ tal que $|Z_{j,k}^{\lambda}(\omega) - Po_{\lambda}(j)| > \epsilon$ para infinitos tales k. Más aún, de acuerdo con [1] existirán λ y ε racionales.

Llamemos $(k_n)_{n \in \mathbb{N}}$ a una tal secuencia infinita que todos cumplan $|Z_{j,k}^{\lambda}(\omega) - Po_{\lambda}(j)| > \epsilon$.

Por arquimedeaneidad, existe m_0 tal que $k_{m_0} > \frac{2}{\epsilon}$ implica $\epsilon > \frac{2}{k_{m_0}}$ implica $\omega \in Bad_{k_{m_0}}$. Por lo tanto, $\forall m \geq m_0$ implica $\omega \in Bad_{k_m}$, y encontramos la secuencia buscada $k_{m_0}, k_{m_0+1}, \ldots$. □

La recíproca no necesariamente es cierta, ya que si bien cuando un string es Poisson-genérico se cumple que $|Z_{j,k}^{\lambda}(\omega) - Po_{\lambda}(j)| \to 0$ para todo $j, \lambda \geq 0$, podría decrecer más lentamente que lo elegido.

Lema 3. *Si existe una secuencia creciente $(k_i)_{i \in \mathbb{N}}$ tal que $\omega \in Bad_{k_n} \forall n$, entonces sucede que* $\limsup_m B(\omega \upharpoonright m) = +\infty$

Demostración. Para todo i, si $m \geq k_i$ entonces $B_{k_i}(\omega \upharpoonright_m) = 1$.

Luego $B(\omega \upharpoonright_m) \geq \sum_{k_i \leq m} B_{k_i}(\omega \upharpoonright_m) = |\{k_i : k_i \leq m\}|$.

Si $m \to \infty$ entonces $B(\omega \upharpoonright_m) \to \infty$, luego $\limsup_m B(\omega \upharpoonright m) = +\infty$. □

3.5. B es computable

Para demostrar que la martingala que definimos es computable daremos un algoritmo que calcula $\hat{B}(\omega, n)$.

3.5.1. Plan

Recordemos que definimos $\hat{B} : \Sigma^* \times \mathbb{N} \to \mathrm{Rat}_2^{\geq 0}$ a una función de aproximación de $B : \Sigma^* \to \mathbb{R}_{\geq 0}$, con precisión 2^{-n}. Es decir, $\hat{B}$ es una función que cumple que para todo $x \in \Sigma^*, n \in \mathbb{N}$, sucede que $|\hat{B}(x, n) - B(x)| < 2^{-n}$.

Vamos a mostrar cómo computar versiones de distintas cantidad de argumentos:

La aproximación de 4 argumentos, $\hat{B} : \mathbb{N} \times \Sigma^* \times \mathbb{Q} \times \mathbb{N} \to \mathrm{Rat}_2^{\geq 0}$, que cumple

$$\left|\hat{B}(k, \omega, \lambda, n) - B_{k,\lambda}(\omega)\right| < 2^{-n}.$$

Luego, ajustando la versión de 4 argumentos, haremos la aproximación de 3 argumentos, $\hat{B} : \mathbb{N} \times \Sigma^* \times \mathbb{N} \to \mathrm{Rat}_2^{\geq 0}$, que cumple

$$\left|\hat{B}(k, \omega, n) - B_k(\omega)\right| < 2^{-n}.$$

Luego veremos cómo combinar la de 3 argumentos para computar la de 2 argumentos.

3.5.2. Algoritmo para computar $\hat{B}$ con 4 argumentos

Primero vamos a analizar el caso donde ω tiene el tamaño justo de la base de los abiertos básicos en $Bad_{k,\lambda}$

Lema 4. *Dado $\omega \in \Sigma^{\lfloor \lambda b^k \rfloor + k}$, para cualquier n, $\hat{B}(k, \omega, \lambda, n)$ es computable en $O(|\omega|)$.*

Demostración. Para computar esta complejidad vamos a tener en cuenta a las operaciones aritméticas, incluida la división, en $O(1)$. Notar que esto genera que n no aparezca en la complejidad de la función de aproximación.

Dado que $B_{k,\lambda}(\omega) = \mu(Bad_{k,\lambda}|\omega)$ y $Bad_{k,\lambda}$ refiere a strings de largo $|\omega|$, esto es equivalente computar la indicadora, es decir ver si $\omega \in Bad_{k,\lambda}$, dado que está en el nivel donde arrancan los abiertos básicos de $Bad_{k,\lambda}$.

Podríamos transformar cada uno de los strings de largo k en ω y armar el histograma.

Una forma más eficiente de hacer esto iterativamente, suponiendo que podemos restar y sumar potencias de b en $O(1)$, es efectivamente iterar de a 1 cada uno de los sub-strings de tamaño k, teniendo en memoria

- Cuántas veces vimos cada una de los $b^k \in O(|\omega|)$ strings posibles,
- Cuántos strings de longitud k vimos $0, 1, \ldots, |\omega|$ veces,
- Cuántos de los $|\omega|$ valores del histograma se alejan de Poisson lo indicado en $Bad_{k,\lambda}$

Una forma de computar esto es mediante el Algoritmo 1 que llamamos IsItBad.

Algorithm 1: IsItBad

Input : ω, λ, k, n
Output: $\omega \in |Bad_k|$

$amounts \leftarrow [0] \cdot b^k$
$histogram \leftarrow [b^k] + [0] \cdot (b^k - 1)$
$badHistogramIdxs \leftarrow |\{i \in 0, b^k, Po_\lambda(i) > \frac{2}{k}\}|$
Function *updateBad*(*amount, i, delta*):
  if $|amount - Po_\lambda(i)| > \frac{2}{k}$ **then**
    $badHistogramIdxs[i] += delta$
Function *updateAmount*(*word, from, to*):
  $updateBad(histogram[from], from, -1)$
  $histogram[from] - -$
  $updateBad(histogram[from], from, 1)$
  $amounts[newRunningString] = to$
  $updateBad(histogram[to], to, -1)$
  $histogram[to] + +$
  $updateBad(histogram[to], to, 1)$
$runningString \leftarrow 0$
for $i \leftarrow 0, \lfloor \lambda b^k \rfloor + k$ **do**
  $next \leftarrow \omega_{size}$
  $newRunningString \leftarrow (runningString * b) + next$
  while $newRunningString \geq b^k$ **do**
    $newRunningString - = b^k$
  if $size \geq k$ **then**
    $updateAmount(runningString, amounts[runningString],$
    amounts[runningString] - 1)
  if $size + 1 \geq k$ **then**
    $updateAmount(newRunningString, amounts[newRunningString],$
    $amounts[newRunningString] + 1)$
return $badHistogramIdxs > 0$

□

Una vez que tenemos esto resuelto, veamos qué sucede cuando ω tiene tamaño menor a los abiertos básicos de $Bad_{k,\lambda}$.

Lema 5. *Dados* $k, \lambda \in \mathbb{R}^+, \omega \in \Sigma^*$, $|\omega| < \lfloor \lambda b^k \rfloor + k$, $\hat{B}(k, \omega, \lambda, n)$ *es computable en* $O(|\omega| + b^{\lambda b^k + k - |\omega|})$.

Demostración. En este caso estamos con un string que podría estar en un nivel de $Bad_{k,\lambda}$ con λ dado, pero queremos analizar cuál es la medida $B_{k,\lambda}(\omega) = \mu(Bad_{k,\lambda}|\omega)$ que corresponde a la cantidad de continuaciones que están en $Bad_{k,\lambda}$. Para eso vamos a tener que analizar todas sus continuaciones.

Entonces podemos usar una leve variante del algoritmo 1, haciendo uso de las funciones y variables ya definidas, de manera similar, usando el código de isItBad y agregando lo siguiente:

Algorithm 2: CalculateBad

Input : $\omega, k_0, k = |\omega|$
Output: $\mu(Bad_{k_0} \cap [\omega])$

Function *howManyBad*(*fillingFromSize*):
- $sum \leftarrow 0$
- **if** $fillingFromSize == \lfloor \lambda b^{k_0} \rfloor$ **then**
  - **return** $badHistogramIdxs > 0?1 : 0$
- $res \leftarrow 0$
- **for** $next \leftarrow 0, \ldots, b-1$ **do**
  - $newRunningString \leftarrow (runningString * b) + next$
  - **while** $newRunningString \geq b^k$ **do**
    - $newRunningString- = b^k$
  - **if** $size \geq k$ **then**
    - $updateAmount(runningString, amounts[runningString],$
    - amounts[runningString] - 1)
  - **if** $size + 1 \geq k$ **then**
    - $updateAmount(newRunningString, amounts[newRunningString],$
    - amounts[newRunningString] + 1)
  - $res \leftarrow res + howManyBad(size + 1)$
- **return** *res*

$IsItBad(\omega)$ //suponiendo que actualiza todas las variables de estado
return $howManyBad(\lfloor \lambda b^k + k \rfloor)/b^{\lfloor \lambda b^{k_0} + k_0 \rfloor}$

La complejidad de este algoritmo va a ser entonces la de probar todas las extensiones de ω de longitud k_0, como fue descripto.

□

3.5.3. Ajustes para computar $\hat{B}$ con 3 argumentos

La diferencia con la versión de 4 argumentos es que varía el λ. Veamos qué cosas hay que ajustarle al código de $\hat{B}$ con 4 argumentos para computar $\hat{B}$ con 3 argumentos.

Recordemos como se define L_k, que a su vez define exactamente qué elementos están en Bad_k.

$$L_k := \{p/q : q \in \{1, \ldots, k\}, p/q < k\}$$

Que varíe λ dentro de los strings de Bad_k nos cambia 2 partes importantes del algoritmo, y llegamos a que

Lema 6. *Dados $k, \lambda \in \mathbb{R}^+, \omega \in \Sigma^*$, $\hat{B}(k, \omega, n)$ es computable en $O(k^2 + log(k)(|\omega| + kb^k))$.*

Demostración. Esta sección necesita considerar 2 partes

Cambios necesarios por la variedad en las longitudes de los strings dentro de Bad_k: las longitudes de los strings en Bad_k no son todas iguales, dado que son de longitud $\lfloor \lambda b^k + k \rfloor$, con $\lambda \in L_k$.

Esto implica que

- Al no tener λ fijo, en vez de iterar hasta $\lfloor \lambda b^k + k \rfloor$, tenemos que iterar hasta el máximo posible, que se puede acotar por $kb^k + k$. Esto hace que las ramificaciones a iterar pasen a ser $b^{kb^k+k-|\omega|}$
- En vez de solo contar la cantidad y dividir por el total al final, hay que ir sumando medidas, y dejar de ramificar cada vez que encontramos un string que está en Bad_k.
- Al iterar, el chequeo de si el string actual está en Bad_k va a hacerse dependiendo de si la longitud del string actual es de la forma $\lfloor \lambda b^k + k \rfloor$, para algún $\lambda \in L_k$.
  - Vamos a tener que tener los λ posibles ordenados. Como son $O(k^2)$, es fácil ordenarlos en $O(k^2 \log(k))$.
  - Los λ se pueden probar a modo de sweep-line: a medida que crece la longitud del string, los λ que cumplen $|\omega| < \lfloor \lambda b^k + k \rfloor$ son menos, con lo que se puede tener en memoria un iterador que indique el último índice en la lista de λ que cumple esto, y los siguientes serán los candidatos a cumplir la igualdad.

Cómo modificar la función updateBad del código de 4 argumentos para tener en memoria la nueva estructura: también cambia lo que hay que tener en memoria para saber si un string está en Bad_k, dado que $Po_\lambda(j)$ depende de λ. Hay que notar que

- Para un j fijo, $\exists \hat{\lambda}$ tal que $Po_\lambda(j)$ es creciente si $\lambda \in (0, \hat{\lambda})$ y decreciente si $\lambda \in (\hat{\lambda}, \infty)$.
- Para x fijo, $|n - x|$ es decreciente con $n \in (-\infty, x)$, y creciente con $n \in (x, \infty)$.

Con estas observaciones podemos notar que, para un j, hay una unión de $O(1)$ intervalos respecto a los índices de λ para los cuales el string está fuera de lo que pide Poisson. También se puede ver que se puede averiguar cuál es la unión de intervalos en $O(\log(k))$.

Entonces vamos a necesitar una estructura de datos que nos de

- Estructura con k^2 lugares, inicializada en todos 0
- Actualización de $O(1)$ intervalos de índices, para hacerles +1 y -1
- Consulta puntual de si un índice tiene valor $= 0$ (o > 0).

Un segment tree nos da estas operaciones, con inicialización $O(k^2)$ y cada una de las operaciones en $O(\log(k))$

En total, esto nos dice que para calcular $\hat{B}$ con 3 argumentos lo vamos a estar haciendo en complejidad temporal

$$O(k^2 + \log(k)(|\omega| + kb^k)).$$

□

3.5.4. Qué B_k podemos descartar sin perder demasiado capital

Vamos a querer acotar $\sum_{k \geq k_0} B_k(\emptyset)$.

Un resultado de [1] nos indica el siguiente lema:

Lema 7. *Para $k \geq 24$, $\mu(Bad_k) \leq 2b^k k^3 \exp\left(\frac{-2b^k}{k^5}\right)$.*

Sea $f(k) = 2b^k k^3 \exp\left(\frac{-2b^k}{k^5}\right)$.

Lema 8. $f(2k) < f(k)\exp(k\ln(8b) - 2b^k)$.

Demostración.

$$\begin{aligned}
f(2k) &= 2b^{2k}(2k)^3 \exp\left(\frac{-2b^{2k}}{(2k)^5}\right) \\
&= 2b^k k^3 b^k 8 \exp\left(\frac{-2b^k}{k^5}\frac{b^k}{2^5}\right) \\
&\leq (2b^k k^3)(8b^k)\exp\left(\frac{-2b^k}{k^5}b^{k-5}\right) \\
&= f(k)\frac{8b^k}{\exp\left(\frac{2b^k}{k^5}(b^{k-5}-1)\right)} \\
&< f(k)\frac{8b^k}{\exp\left(2b^k\right)} \\
&= f(k)\exp(\ln(8b^k) - 2b^k) \\
&= f(k)\exp(k\ln(8b) - 2b^k).
\end{aligned}$$

□

Sea $g(k) = 2b^k - k\ln(8b)$.

Lema 9. *Con $b \geq 2, k \geq 2$, si $g(k) > 2^k$ entonces $g(k+1) > 2^{k+1}$.*

Demostración.

$$\begin{aligned} g(k+1) &= 2b^{k+1} - (k+1)ln(8b) \\ &= g(k) - ln(8b) + 2b^k(b-1) \\ &> 2^k + 2b^k\left(b - \frac{3}{2}\right) \\ &> 2^{k+1} \end{aligned}$$

□

Corolario 1. $f(2k) < f(k)\frac{1}{2^{2^k}}$.

Derivando f nos queda

$$2k^3 e^{-\frac{2b^k}{k^5}} b^k \left(\log(b) + \frac{3}{k} + \frac{10b^k}{k^6} - \frac{2b^k \log(b)}{k^5}\right).$$

Notemos que esta expresión, con $k > 0$, va a ser negativa cuando

$$\begin{aligned} \log(b) + \frac{3}{k} + \quad & \frac{10 \cdot b^k}{k^6} - \quad & \frac{2 \cdot b^k \log(b)}{k^5} < 0 \longleftrightarrow \\ k^6 \log(b) + 3k^5 + \quad & 10 \cdot b^k - \quad & 2 \cdot b^k \log(b)k < 0 \leftrightarrow \\ (k^6 \log(b) + 3k^5 - b^k) + \quad & (10 \cdot b^k - b^k \cdot k) - \quad & b^k(2k\log(b) - (k+1)) < 0. \end{aligned}$$

Ahora bien, mirando los 3 componentes de esta suma por separado,
Llamando

$$\begin{aligned} l_1(k) &= k^6 \log(b) + 3k^5 - b^k \\ l_2(k) &= 10 \cdot b^k - b^k \cdot k \\ l_3(k) &= -b^k(2k\log(b) - (k+1)) \end{aligned}$$

Observación 8. *$l_1(k) < 0$ para $k > 30$ y $b \geq 2$.*

Demostración.

$$\begin{aligned} &l_1(k+1) < 2bk^6 - b^k = b^{6\log_b(k)+1+\log_b(2)} - b^k < 0 \leftrightarrow \\ &6\log_b(k) + 1 + \log_b(2) - k < 0 \end{aligned}$$

Si $g(k) = 6\log_b(k) + 1 + \log_b(2) - k$,

- $g(30) < 0$ para $b = 2$, y g es decreciente al crecer b.
- $g'(k) = \frac{6}{k\log(b)} - 1$, por lo que si $k > 8 > \frac{6}{\log(2)} - 1 \geq \frac{6}{\log(b)} - 1$ sabemos que tiene derivada negativa.

Luego se cumple $l_1(k) < 0$ para $k > 30$. □

Observación 9. *$l_2(k) < 0$ para $k > 30$,*

Demostración.

$$10 \cdot b^k - b^k \cdot k < 0 \leftrightarrow$$
$$10 - k < 0, \text{Y esto sucede para } k > 10.$$

□

Observación 10. *$l_3(k) < 0$ para $k > 30$*

Demostración.

$$-b^k(2k\log(b) - (k+1)) < 0 \leftrightarrow$$
$$2k\log(b) - (k+1) > 0 \leftrightarrow$$

Esto sucede para $k = 5$ ya es positivo, y tiene derivada constante positiva. □

Por ende, f es decreciente para $k > 30$.

Lema 10. $\sum_{k \geq k_0} B_k(\emptyset) < \frac{1}{2^{2^{k_0-1}}}$.

Demostración.

$$\begin{aligned}
\sum_{k \geq k_0} B_k(\emptyset) &= \sum_{k \geq k_0} \mu(Bad_k) \\
&\leq \sum_{k \geq k_0} f(k) \\
&< f(k_0) \sum_{lk \geq \log_2(k_0)} \frac{2^{lk}}{2^{2^{lk}}} \\
&< f(k_0) \frac{1}{2^{2^{k_0-1}}} \\
&< \frac{1}{2^{2^{k_0-1}}}
\end{aligned}$$

□

3.5.5. Cómo computar $\hat{B}$ de 2 argumentos

Demostración del teorema 3.1. Por lo que ya vimos,

$$\sum_{k_0 \geq \log_2(n)+2} B_{k_0}(\omega) < \frac{1}{2^{2^{\log_2(n)+2-1}}} = \frac{1}{2^{2n}} < \frac{1}{2^{n+1}}$$

para $n \geq 1$,

Si computamos $\hat{B}(\omega, n)$ de manera que

$$\left| \hat{B}(\omega, n) - \sum_{k_0=1}^{\log_2(n)+2} B_{k_0}(\omega) \right| < 2^{-(n+1)}$$

conseguimos 2^{-n} de error respecto a $B(\omega)$ ignorando los demás términos que forman parte.

Entonces queremos calcular $\sum_{k_0=1}^{\log_2(n)+2} B_{k_0}(\omega)$. Para esto calculamos cada uno de los $B_{k_0}(\omega)$ usando $\hat{B}$ de 3 argumentos, recordando que tenemos precisión exacta suponiendo aritmética perfecta en $O(1)$.

Sumando la complejidad de $\hat{B}$ de 3 argumentos sobre $k_0 \leq \log_2(n) + 2$, nos queda que la complejidad de esta parte del algoritmo es

$$O(\log(n)^3 + \log(n)^2|\omega| + \log\log(n)(2^{\log(n)(2^{\log(n)}+1)-|\omega|})).$$

Ajustando estas cuentas al modelo estandar de complejidad, nos queda que la complejidad del algoritmo final es

$$O(|n|^3 + |n|^2|\omega| + \log(|n|)(2^{|n|(2^{|n|}+1)-|\omega|})) =$$
$$O((2^{|n|(2^{|n|}+1)-|\omega|})\log(|n|) + |n|^2(|n| + |\omega|)).$$

Dado que la complejidad de una martingala se calcula en base a la longitud de su input, que es $m = |\omega| + |n|$, vemos que

$$2^{m2^m} = 2^{(|n|+|\omega|)(2^{|n|+|\omega|})} = 2^{|n|2^{|n|}2^{|\omega|}} 2^{|\omega|2^{|n|}2^{|\omega|}}.$$

Para $|\omega| > 0$ esto es mayor que

$$(2^{|n|(2^{|n|}+1)-|\omega|})\log(|n|),$$

que es la complejidad de la función, con lo que la complejidad de la función nos queda dentro de $O(2^{m2^m})$, como buscábamos. □

Un detalle interesante a remarcar es que, como la complejidad es relativamente baja respecto a $|\omega|$, pedir que n esté escrito en unario en el input nos cambiaría considerablemente la complejidad. El unario es un concepto relevante en teoría de la complejidad, y consistiría en escribir la precisión n como parte del input como tantos 1 como su valor. Esto enfatizaría el hecho de que la complejidad, vista desde el punto de vista de $|\omega|$, es relativamente baja.

4. CONJETURAS BASADAS EN INDEPENDENCIA SEGÚN POISSON

La idea de la siguiente conjetura va a ser poder acotar la medida de cualquier conjunto Bad que esté contenido en un string prefijo que no este en el Bad con abiertos básicos de su tamaño, por algo que decrezca exponencialmente.

En esta sección vamos a trabajar con $b = 2, \lambda = 1$, dado que es una idea aún a explorar y solo trabajada en esa base. El cambio de base no es sustancial, pero arrastra más términos en las cuentas.

Conjetura 1. $\exists \varepsilon > 0, f : \mathbb{N} \to \mathbb{R}$ *tal que*

$\text{máx}\left(\{\mu(Bad_{n+1} \cap [\omega]) : \omega \in \Sigma^{2^n+n}, \omega \notin Bad_n\}\right) < f(n)$, *y*

$\lim \sum \log\left(1 - (1+\varepsilon)\frac{f(n)}{1-f(n)}\right)$ *converge.*

La esperanza detrás de esta conjetura es formalizar que si ω no es un string malo, la mayoría de sus continuaciones tampoco deberían serlo. Hay que revisar si ese máx es muy estricto, quizás se puede pedir algo un poco más débil.

La idea de que esto está basado en la independencia es que la medida buscada debería ser parecida a la medida de Bad_n, excepto por strings fuera de Bad_n que no sean independientes con ω. Intuitivamente, como hay pocos strings en Bad_{n+1}, uno esperaría que si la primera mitad del string no estaba en Bad_n, lo más probable sea que el string completo no esté en Bad_{n+1}, salvo que haya correlación entre ambas mitades.

El límite está armado específico para probar la siguiente martingala, pero se podría pedir algo como que la medida decrezca supraexponencialmente y debería estar implicando dicho límite.

Conjetura 2. *Si vale la Conjetura 1, existe una martingala computable para los strings que no son Poisson genéricos con complejidad lineal en* $|\omega| + n$

La martingala en cuestión va a ser la suma de 2 martingalas. Una martingala va a atrapar a los strings infinitos que tenga todos sus prefijos a partir de cierto tamaño en Bad_k para $k \geq k_0$, y la otra a aquellos que no.

Conjetura 3. *Existe una martingala lineal B que tiene éxito en los strings ω en las que $\exists n_0$ tal que para todo $n \in \mathbb{N}$, $n \geq n_0 \to \omega \in Bad_n$*

Una martingala que cumple esto es $B = \sum B_{f(n)}$, con B_k la martingala que atrapa los strings en Bad_k, y $f(n)$ tal que $\sum_{k > \log_2^2(n)} B_{f(k)}(\emptyset) < 2^{-(n+1)}$.

Dicha f existe, ya que $\sum B_k < \infty$.

La martingala es efectiva en dichos strings, ya que $\exists n_0$ tal que para todo $n \geq n_0$ ocurre que $B_{f(n)}(\omega \upharpoonright m) \underset{m\to\infty}{\to} 1$,

En este caso, la restricción de f nos ayuda para calcular $\hat{B}(\omega, n)$, ya que podemos ignorar los términos $B_{f(m)}$ con $m > \log_2^2(n)$, con lo que vamos a tener que calcular con precisión $2^{-(n+1)}$ la suma de las martingalitas $B_1, \ldots, B_{\log_2^2(n)}$, que vimos cómo calcular de manera exacta.

Ahora, para calcular $B_{\log_2^2(n)}(\omega)$ hay que llamar a CalculateBad, dando una complejidad de $O(2^{2^{\log_2^2(n)-k}})$

Con algunos ajustes extra, pensamos que esto se puede dejar con complejidad $O(n)$.

Conjetura 4. *Existe una martingala lineal B que podemos computar que tiene éxito en los strings ω que tienen infinitos índices $k \in \mathbb{N}$ para los que $\omega \in Bad_k$ e infinitos para los que $\omega \notin Bad_k$*

Una posible martingala es de la siguiente forma:

$$B(\omega_1\omega_2) = \begin{cases} B(\omega_1) & \text{si } \omega_1 \in Bad_{k-1} \\ WB(\omega_1) & \text{sino, y } \omega_1\omega_2 \in Bad_k \\ \left(1 - \frac{W|Bad_k \cap [\omega_1]|}{|(Bad_k \cap [\omega_1])^C|}\right) B(\omega_1) & \text{sino} \end{cases} \tag{4.1}$$

Tomando el $\epsilon > 0$ de 1, usamos $W = 1 + \varepsilon$, y luego la proporción que se pierde está acotada por debajo por la productoria de la conjetura, pero lo que se gana no está acotado, ya que $W > 1$.

En cuanto a la complejidad de calcular $\hat{B}(\omega, n)$, esperamos que sea lineal, porque al haber cota inferior de la pérdida, el término que va a dominar en dicha cuenta es W^n.

Una posible definición de independencia podría aprovechar $b = 2$ y usar el xor ($\wedge$) bit a bit para combinar secuencias, es decir, para todo $a, b \in \Sigma^* \cup \Sigma^{\mathbb{N}}$ y $n \in \mathbb{N}$,

$$(a \wedge b) \upharpoonright n = (a \upharpoonright n) \wedge (b \upharpoonright n),$$

donde el i-ésimo dígito será 1 si y solo si exactamente 1 de los i-ésimos dígitos entre $a \upharpoonright n$ y $b \upharpoonright n$ es 1.

Una posibilidad para explorar cotas similares e independencia viene explorando la posibilidad de combinar distintos strings, por ejemplo

Definición 14. *2 strings ω, τ son **independientes según Poisson** si*

$$(\omega \wedge \tau) = (\omega_1 \wedge \tau_1)(\omega_2 \wedge \tau_2)\dots$$

es Poisson genérica

Análogamente podemos definir

Definición 15. *2 strings finitos $\omega, \tau \in \Sigma^{2^k}$ son **dependientes de orden k según Poisson** si $\omega \wedge \tau \in Bad_k$*

Lo bonito de esta definición es que los strings dependientes de orden k son pocas, por lo que podríamos intentar apostar a los sufijos que están en Bad o a la dependencia de orden k. Para esto, necesitaríamos que sea válida la siguiente conjetura, que reforzaría la idea de que concatenar strings independientes según Poisson nos da una secuencia Poisson genérica.

Conjetura 5. *Si tenemos $\omega \in \Sigma^{2^k}$, y tenemos una secuencia de strings $(\tau_n)_{n\in\mathbb{N}}$ de manera que para todo m, $\tau_m \in \Sigma^{2^{k-1+m}}$ es tal que τ_m es independiente de orden $k-1+m$ con $\omega\tau_1\tau_2\dots\tau_{m-1}$, luego $\omega\tau_1\tau_2\dots$ es Poisson genérica.*

5. CONCLUSIONES Y TRABAJO FUTURO

En este trabajo nos dedicamos a analizar distintas opciones de martingalas para capturar a los string que no son Poisson genéricos.

Por un lado, fuimos por un camino bastante marcado, guiados por el trabajo de [1] y [7]. Eso nos dio un resultado inicial.

Explorar distintas estrategias "golosas" y adaptarlas al problema en cuestión nos generó una martingala distinta, levemente más simple, pero que terminó siendo muy similar a la martingala del camino original, con alguna simplificación.

También tuvimos algunos resultados negativos, marcando que ciertas estrategias no funcionan, en base a que no logran capturar la ausencia de este tipo de azar.

El trabajo futuro esencial parece ser investigar más sobre los strings que no son Poisson genéricos. Entender cómo funcionan distintas operaciones sobre ellas, con algunas conjeturas citadas en este trabajo.

Por la naturaleza binomial/Poisson descripta en [1], es interesante estudiar más sobre resultados acerca de procesos de Poisson y cuál puede ser su relación con estos strings. Un ejemplo concreto de esto es la noción de independencia mencionada superficialmente en este trabajo.

En un enfoque radicalmente distinto, otra opción poco explorada fue la de calcular los string *Bad* de manera combinatoria, en busca de no tener que analizar todas los strings para generar un histograma. Surgen problemas de grafos interesantes, aunque también intuimos que es podría surgir un resultado de teoría de complejidad en vez de una solución de mejor complejidad al problema.

6. APÉNDICE: ALGUNAS OBSERVACIONES ADICIONALES

Observación 11. *Con el test enunciado en [1] es posible armar una martingala computable.*

El paper original propone un Test de Martin-Lof que al ser computable es un Test de Schnorr.

Tomando este test y mirando las martingalas que apuntan a las distintas uniones infinitas de niveles del test, que a su vez son uniones infinitas de strings malos, tenemos una alternativa para una martingala computable. Esperamos que tenga mayor complejidad que la descripta.

Observación 12. *No existe martingala que gane capital cada vez que un string está en Bad_k y tenga éxito en strings que no son Poisson genéricos.*

Se puede ver que existiría un $W > 1$ tal que, con $|\omega_1| = |\omega_2| = 2^k + k$, $B(\omega_1\omega_2) \geq WB(\omega_1)$.

Observando el string $0^{\mathbb{N}}$ se puede llegar a esta observación, ya que muchas de las continuaciones de longitud $2^{k+1} + k + 1$ de 0^{2^k+k} estarán en Bad_{k+1}.

Observación 13. *No se puede definir una martingala que solo apueste sobre futuro, del mismo tamaño que los dígitos ya vistos, y tenga éxito sobre los strings que no son Poisson genéricos*

Esto soluciona alguno de los problemas de la idea anterior, pero no funciona ya que

Observación 14. *Que un string ω cumpla que $\exists n_0$ tal que para todo $n \geq n_0$ sucede que $\omega[2^n, 2^{n+1}] \notin Bad_n$ no alcanza para decir que es Poisson genérica.*

Supongamos que esto no fuera cierto, y que esa implicación nos diera un string Poisson genérica. Entonces tomemos un string ω que sea Poisson genérico, y generemos un string que cumple las mismas condiciones y no es Poisson genérico, τ, de la siguiente forma:

Sea n_1 el primer número tal que $n_1 \geq n_0$ y $\omega[1, ..., 2^{n_1}] \notin Bad_{n_1}$

Pongamos $\tau[1, \ldots, 2^{n_1}] = \tau[2^{n_1} + 1, \ldots, 2^{n_1+1}] = \omega[1, \ldots, 2^{n_1}]$

Ahora bien, supusimos que valía la conjetura, con lo que si completáramos τ con lo que sigue de ω debería ser Poisson genérica. Por ende, existe un n_2 tal que para todo $n \geq n_2$ sucede que $\tau[1, \ldots, 2^{n_1+1}]\omega[2^{n_1+1} + 1, \ldots, 2^n] \notin Bad_n$

Ahora pongamos $\tau[2^{n_1+1}+1, \ldots, 2^n] = \omega[2^{n_1+1}+1, \ldots, 2^n]$, y $\tau[2^n+1, \ldots, 2^{n+1}] = \tau[1, \ldots, 2^n]$

Sigamos este procedimiento para $n_3, n_4, \ldots$

Es claro que τ cumple la premisa, dado que $\tau[2^n, 2^{n+1}]$ es, o bien $\omega[2^n, 2^{n+1}]$, que lo definimos de manera que cumpla, o bien es igual a $\tau[1, 2^{n_i}]$, que también lo definimos de manera que cumpla.

Por otro lado, por lo visto previamente, $\tau[1, 2^{n_i}]\tau[1, 2^{n_i}] = \tau[1, 2^{n_i}]\tau[2^{n_i}+1, 2^{n_i+1}] = \tau[1, 2^{n_i+1}] \in Bad_{n_i+1}$, por lo que τ no es Poisson genérica. Llegamos a un absurdo que demuestra lo propuesto.

Bibliografía

[1] Nicolás Álvarez, Verónica Becher, Martín Mereb. Poisson generic Sequences. International Mathematics Research Notices rnac234, 2022.

[2] Verónica Becher, Sac Himelfarb. A construction of a λ-Poisson generic sequence. Mathematics of Computation 92:1453-1466, 2023.

[3] Verónica Becher, Stephen Jackson, Dominik Kwietniak, Bill Mance. The Descriptive Complexity of the Set of Poisson Generic Numbers, Journal od Mathemathical Logic, en prensa 2024.

[4] Lauren Bienvenu, Glenn Shafer, Alexander Shen. Martingales in the Study of Randomness. In: Mazliak, L., Shafer, G. (eds) *The Splendors and Miseries of Martingales*. Trends in the History of Science. Birkhäuser, Cham, 2022.

[5] Rod Downey and Denis Hirschfeldt. *Algorithmic randomness and complexity*. Springer, New York, 2010.

[6] Santiago Figueira, André Nies, Feasible Analysis, randomness, and base invariance, Theory of Computing Systems 56: 439–464, 2015.

[7] André Nies. *Computability and Randomness*, Clarendon Press, Oxford, 2008.

[8] Benjamin Weiss. Poisson generic points, 23-27 November 2020. Conference Diophantine Problems, Determinism and Randomness - CIRM- organized by Joël Rivat and Robert Tichy. `https://library.cirm-math.fr/Record.htm?idlist=13&record=19287810124910050929`