Departamento de Computación, Facultad de Ciencias Exactas y Naturales, UBA

Azar y Autómatas

Clase 8: Aleatoriedad pura (complejidad Chaitin-Kolmogorov)

Pure randomness

A sequence is random if its initial segments can only be described explicitely by a Turing machine. That is, its initial segments cannot be compressed with a Turing machine.

Formally, a sequence is random if its initial segments have almost maximal program-size complexity.

Kolmogorov / program-size complexity

Some long strings can be described using fewer symbols than their length; this is used in data compression.

For example, string consisting of 2^n many a's can be encoded as $\log n$ many symbols plus a constant:

```
input n
i=0;
while (i<2^n) {print a; i=i+1;}
```

Kolmogorov / program-size complexity

Fix a universal Turing machine $U : \{0,1\}^* \to \{0,1\}^*$.
Kolmogorov (1965) defined the complexity of a string $s \in \{0,1\}^*$ as the length of the shortest input in U that outputs s.

Kolmogorov / program-size complexity

Fix a universal Turing machine $U : \{0,1\}^* \to \{0,1\}^*$.
Kolmogorov (1965) defined the complexity of a string $s \in \{0,1\}^*$ as the length of the shortest input in U that outputs s.

Definition (Chaitin 1975)

Fix a Turing machine U with prefix-free domain. The program-size complexity is a function $K_U : \{0,1\}^* \to \mathbb{N}$ such that

$$K_U(s) = \min\{|p| : U(p) = s\}$$

Kolmogorov / program-size complexity

Fix a universal Turing machine $U : \{0,1\}^* \to \{0,1\}^*$.
Kolmogorov (1965) defined the complexity of a string $s \in \{0,1\}^*$ as the length of the shortest input in U that outputs s.

Definition (Chaitin 1975)

Fix a Turing machine U with prefix-free domain. The program-size complexity is a function $K_U : \{0,1\}^* \to \mathbb{N}$ such that

$$K_U(s) = \min\{|p| : U(p) = s\}$$

A machine U is optimal if, for all Turing machines M with prefix free domain, $K_U \leq K_M + constant_M$.

Kolmogorov / program-size complexity

Fix a universal Turing machine $U : \{0,1\}^* \to \{0,1\}^*$.
Kolmogorov (1965) defined the complexity of a string $s \in \{0,1\}^*$ as the length of the shortest input in U that outputs s.

Definition (Chaitin 1975)

Fix a Turing machine U with prefix-free domain. The program-size complexity is a function $K_U : \{0,1\}^* \to \mathbb{N}$ such that

$$K_U(s) = \min\{|p| : U(p) = s\}$$

A machine U is optimal if, for all Turing machines M with prefix free domain, $K_U \leq K_M + constant_M$.

Consider any optimal machine U. Drop the subindex and write K.

Obvious properties of K

Let $U : \{0,1\}^* \to \{0,1\}^*$ be an optimal Turing machine U with prefix-free domain.
For every string s, t,

- $K(s) \leq |s| + 2 \log |s| + constant$.
- $K(st) \leq K(s) + K(t) + constant$
- If f is computable then $K(f(s)) \leq K(s) + constant_f$.

The Berry's paradox

Give the smallest positive integer not definable in fewer than thirteen words.

The Berry's paradox

Give the smallest positive integer not definable in fewer than thirteen words.
The above sentence has twelve.

The Berry's paradox

Give the smallest positive integer not definable in fewer than thirteen words.
The above sentence has twelve.

G.G. Berry 1867–1928, librarian at Oxford's Bodleian library.

G. Boolos (1989) built on a formalized version of Berry's paradox to prove Gödel's Incompleteness Theorem formalizing the expression "m is the first number not definable in less than k symbols".

X. Caicedo (1993), La paradoja de Berry revisitada, o la indefinibilidad de la definibilidad y las limitaciones de los formalismos Lecturas Matemáticas 14: 37-48.

The Berry's paradox

Give the smallest positive integer not definable in fewer than thirteen words.
The above sentence has twelve.

G.G. Berry 1867–1928, librarian at Oxford's Bodleian library.

G. Boolos (1989) built on a formalized version of Berry's paradox to prove Gödel's Incompleteness Theorem formalizing the expression "m is the first number not definable in less than k symbols".

X. Caicedo (1993), La paradoja de Berry revisitada, o la indefinibilidad de la definibilidad y las limitaciones de los formalismos Lecturas Matemáticas 14: 37-48.

Though the formal analogue does not lead to a logical contradiction, it yields a proof that Kolmogorov complexity K is not computable.

Randomness ♥ Logic

Theorem

Let U be a universal Turing machine. The function $K : \{0,1\}^* \to \mathbb{N}$, $K(s) = \min\{|t| : U(t) = s\}$ is not computable.

Proof. Assume K is computable. Consider the following program:

```
int main(){
    int K(String s){ ....}
```

Randomness ♥ Logic

Theorem

Let U be a universal Turing machine. The function $K : \{0,1\}^* \to \mathbb{N}$, $K(s) = \min\{|t| : U(t) = s\}$ is not computable.

Proof. Assume K is computable. Consider the following program:

```
int main(){
    int K(String s){ ....}
    const C = 10000; /* greater than or equal to this program length*/
```

Randomness ♥ Logic

Theorem

Let U be a universal Turing machine. The function $K : \{0,1\}^* \to \mathbb{N}$, $K(s) = \min\{|t| : U(t) = s\}$ is not computable.

Proof. Assume K is computable. Consider the following program:

```
int main(){
    int K(String s){ ....}
    const C = 10000; /* greater than or equal to this program length*/
    String s=empty word;
    while (K(s) ≤ C) s= next(s);
    print s;
```

Randomness ♥ Logic

Theorem

Let U be a universal Turing machine. The function $K : \{0,1\}^* \to \mathbb{N}$, $K(s) = \min\{|t| : U(t) = s\}$ is not computable.

Proof. Assume K is computable. Consider the following program:

```
int main(){
    int K(String s){ ....}
    const C = 10000; /* greater than or equal to this program length*/
    String s=empty word;
    while (K(s) ≤ C) s= next(s);
    print s;
}
```

According to the execution $K(output) > C$.
However,

Randomness ♥ Logic

Theorem

Let U be a universal Turing machine. The function $K : \{0,1\}^* \to \mathbb{N}$, $K(s) = \min\{|t| : U(t) = s\}$ is not computable.

Proof. Assume K is computable. Consider the following program:

```
int main(){
    int K(String s){ ....}
    const C = 10000; /* greater than or equal to this program length*/
    String s=empty word;
    while (K(s) ≤ C) s= next(s);
    print s;
}
```

According to the execution $K(output) > C$.
However, $K(output) \leq |\texttt{int main()\{...\}}| \leq C$.
Contradiction.

Ejemplo de casi máxima complejidad

Definition (Programa elegante)

Decimos que $p \in \{0,1\}^*$ es un programa elegante si existe $s \in \{0,1\}^*$, $U(p) = s$ y para todo $p' \in \{0,1\}^*$, si $U(p') = s$ entonces $|p| \leq |p'|$.

Llamamos s^* al programa elegante que computa s, es decir $U(s^*) = s$ y $K(s) = |s^*|$.

Proposition

Los programas elegantes tienen casi máxima complejidad K.

Demostración de la complejidad programas elegantes

Sabemos que para toda cadena existe un programa elegante que la retorna. Supongamos ahora que p es un programa. entonces $p*$ es el programa elegante que retorna p.
Sea $b_1 b_2 \dots b_c p^*$ un programa que

1. lee $p*$,
2. obtiene p,
3. computa p y
4. output s.

Es decir, $U(b_1 b_2 \dots b_c p*) = s$.
Como p es elegante debe cumplirse que

$$|p| \leq |b_1 b_2 \dots b_c p^*| = c + |p^*| = c + K(p)$$

Concluímos que

$$K(p) \geq |p| - c$$

Notemos que la constante c es independiente de p y por lo tanto los programas elegantes son todos incompresibles respecto de la misma constante c.

La mayoría de las palabras son incompresibles

Decimos que la cadena s es compresible para Δ si y solo si $K(s) < |s| + K(|s|) + \Delta$.

Proposition

Dado Δ, la mayoría de las palabras son incompresibles para Δ.

Sea M la cantidad de cadenas de largo n incompresibles para Δ. Hay exactamente 2^n cadenas de largo n. Luego $0 \leq M \leq 2^n$. Expresaremos M como una cadena de n bits, rellenando con ceros a la izquierda en caso de ser necesario. Demostraremos que $K(M) > n - \textit{constante}$. Esto significa que M no puede tener demasiados ceros a la izquierda, ya que en este caso M sería compresible.

Una herramienta más

Necesitamos una definición más.
Sea s^* la cadena tal que $U(s^*) = s$ y $|s^*| = K(s)$

$$K(s/t) = \min\{p : U(p, t^*) = s\}$$

Chaitin demuestra

$$K(s, t) \leq K(s/t) + K(t) + O(1)$$

La cantidad de cadenas incompresibles para Δ

Sea M la cantidad de cadenas incompresibles para Δ de largo n.
Sea el siguiente programa que recibe tres datos de entrada:
- n^*, el programa mínimo para n,
- Δ, diferencia entre $n + K(n)$ y complejidad máxima de cadenas de n bits
- q, un programa mínimo para M dado n^*.

El programa hace esto:

1. Calcula $K(n)$.
2. Computa n.
3. Computa M.
4. Encuentra $2^n - M$ cadenas compresibles de largo n. Esto se realiza ejecutando todos los programas de longitud menor que $n + \Delta$ hasta encontrar $2^n - M$ de ellos que se detienen y quedarse con su output.
5. Imprime x la primer cadena incompresible de largo n, según el orden lexicográfico. Notar que si Δ es un número negativo entonces hay a lo sumo $2^{(n+\Delta)} - 1$ cadenas compresibles, y por lo tanto hay certeza de encontrar cadenas incompresibles.)
6. Termina.

Sabemos que la complejidad de la cadena x retornada por el programa es

$$K(x) \geq n + K(n) + \Delta$$

Por otro lado obtuvimos x con este programa p que recibe n^*, Δ y q.

$$K(x) \leq K(x/n, \Delta, q) + K(n, \Delta, q) \leq |p| + K(n) + \Delta + K(M/n)$$

Poniendo las dos desigualdades juntas

$$n + K(n) + \Delta \leq |p| + K(n) + \Delta + K(M/n)$$

Entonces

$$n \leq |p| + K(M/n)$$

$$K(M/n) > n - |p|$$

$$K(M) > n - p$$

Para valores grandes de n la longitud del progama p se hace despreciable, y por lo tanto para expresar M dado n^* hacen falta todos los n bits. $\square$

Probabilidades y comportamiento de computadora

Definition (Chiaitin 1975)

Sea $M : \{0,1\}^* \to \{0,1\}^*$ función recursiva parcial con dominio libre de prefijo. For $s \in \{0,1\}^*$,

$$P_M(s) = \sum_{p \text{ such that } M(p)=s} 2^{-|p|}$$

Cuando M es universal U que fijamos no usamos el subindice.
Thus, we must show that $0 \leq P(s) \leq 1$.

Probabilidades y comportamiento de computadora

Definition (Chiaitin 1975)

Sea $M : \{0,1\}^* \to \{0,1\}^*$ función recursiva parcial con dominio libre de prefijo. For $s \in \{0,1\}^*$,

$$P_M(s) = \sum_{p \text{ such that } M(p)=s} 2^{-|p|}$$

Cuando M es universal U que fijamos no usamos el subindice.
Thus, we must show that $0 \leq P(s) \leq 1$.

Probabilidades y comportamiento de computadora

Theorem (Desigualdad de Kraft)

Si L es un conjunto libre de prefijos entonces

$$\sum_{w \in L} 2^{-|w|} \leq 1.$$

La demostración interpreta geométrica un lenguaje $L \subseteq \sum^*$.
Consideremos el intervalo $[0, 1]$ de la recta real. La prueba consiste en mostrar que si asignamos a cada palabra de L un subintervalo de $[0, 1]$ de longitud $2^-|w|$ entonces, si L es libre de prefijos los intervalos asignados no se superponene entre sí y por lo tanto la suma de las longitudes de los subintervalos asignados es menor que 1.

Demostración de la desigualdad de Kraft

Para cada longitud n dividimos el $[0,1]$ en 2^{-n} subintervalos.

Sabemos que en $0,1^*$ hay exactamente 2^n palabras de longitud n; si consideramos las 2^n palabras de longitud n ordenadas lexicográficamente nos permite asociar de manera natural a cada palabra con un ordinal k.

Por ejemplo para $w = 000$, $k = 0$ ya que 000 es la primera palabra de longitud 3 mientras que 100 es la quinta $(k = 4)$.
Asignamos a cada palabra $w \in L$ un subintervalo de $[0,1]$, mediante la siguiente función. Si w es la k-ésima palabra en el orden lexicográfico restringido a las palabras de la misma longitud que w,

$$f(w) = [k2^{-|w|}, (k+1)2^{-|w|}]$$

Demostración de la desigualdad de Kraft, continuación

La función f asigna intervalos que son disjuntos o se incluyen unos a otros: para todo v, w

$f(w) \subseteq f(v)$ o $f(w) \subseteq f(v)$ o $f(v) \cap f(w) = \emptyset$.

Sean v, w dos palabras arbitrarias en $0, 1^*$ tales que sus intervalos se intersecan propiamente. Supongamos $|v| = m$ y $|w| = n$. Entonces

$$f(v) = [k2^{-m}, (k+1)2^{-m}] \quad y \quad f(w) = [\ell 2^{-n}, (\ell+1)2^{-n}]$$

Como los intervalos tienen intersección no vacía tenemos o bien

1. $k2^{-m} < \ell 2^{-n} < (k+1)2^{-m} < (\ell+1)2^{-n}$
2. $\ell 2^{-n} < k2^{-m} < (\ell+1)2^{-n} < (k+1)2^{-m}$.

Si $n > m$ llegamos a estas contradicciones:

$$k2^{-m+n} < \ell < (k+1)2^{-m+n} < (\ell+1)$$

$$\ell < k2^{-m+n} < (\ell+1) < (k+1)2^{-m+n}.$$

El caso $m > n$ es análogo. $\square$

Pogram-size complexity "dual to " probability

Theorem (Chaitin 1975)

There is a constant c, for any string s, $2^{-K(s)} \leq P(s) \leq c2^{-K(s)}$.

Pogram-size complexity "dual to " probability

Theorem (Chaitin 1975)

There is a constant c, for any string s, $2^{-K(s)} \leq P(s) \leq c2^{-K(s)}$.

Shannon's entropy is essentially expected program-size complexity:

$$\sum_s P(s)(-\log P(s)) \simeq \sum_s P(s)K(s).$$

Definition (Shannon 1948)

Given a probability P of a discrete random variable X, the entropy $H(X) = \sum_x P(x = X)(-\log P(x = X)))$.

Twins

Program-size complexity is formally identical to Shannon's Information Theory

Photo: Diane Arbus

Program size complexity and infinite sequences

1. Suppose x is computable. Then there is C such that for every $i \geq 1$, $K(x[1..i] \leq K(i) + c$
2. Asumamos una enumeración fija de todas las máquinas de Turing, $TM_1, TM_2, TM_3, \ldots$. Sea $A : a_1 a_2 a_3 \ldots$ donde $a_i = 1$ si y solo si la i-esima máquina de Turing con la cinta vaciía se detiene.
   Supongamos j la cantidad de máquinas de Turing que se detienen con la cinta vacía entre las primeras i máquinas. Por lo tanto, $j \leq i$.

$$\forall i \, K(A[1..i]) \leq K(A[1..i]/i, j) + K(i, j) + \textit{constante}$$
$$K(i, j) \leq K(i) + K(j) + \textit{constante}$$
$$K(i) \leq 2 \ \textit{log} \ i + \textit{constante}$$
$$K(j) \leq 2 \ \textit{log} \ j + \leq 2 \ \textit{log} \ i + \textit{constante}$$

   Notemos que $K(A[1..i]/i, j)$ es una constante, es la longitud del pregrama más corto que realiza "dovetailing" entre las primeras i máquinas hasta encontrar las j que se detienen y así consigue cada uno de los bits de $A[1..i]$. Luego,

$$\forall i \ K(A[1..i]) \leq 4 \ \textit{log} \ i + \textit{constante}$$

The definition of randomness

Definition (Chaitin 1975)

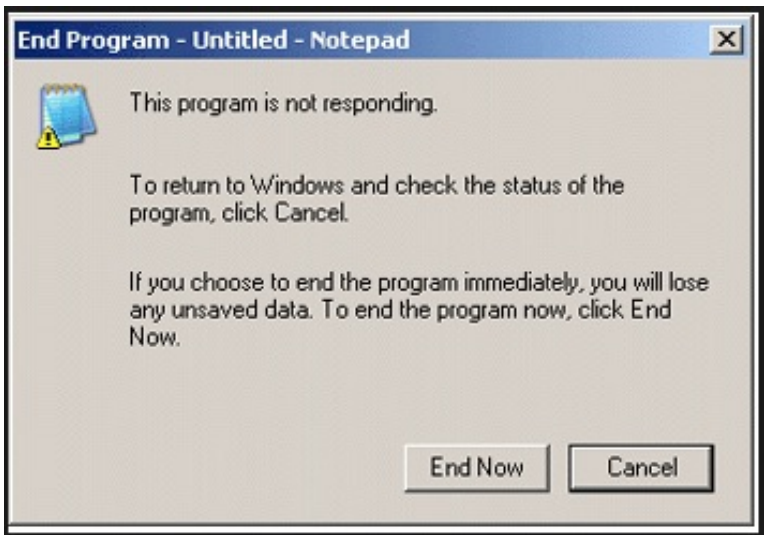
An infinite sequence $x \in \{0,1\}^\omega$ is random if $\exists c \; \forall n \; K(x[1..n])) > n - c$.

The definition applies immediately to real numbers (one-to-one correspondence between reals and their expansions in any given base).

Claramente las secuencias computables no son aleatrias
Y hay secuencias no computables que no son aleatorias.

Examples of random sequences

Have you ever experienced that your computer locked up (froze)?



Ω-numbers

Theorem (Chaitin 1975)

The probability that a universal Turing machine with prefix-free domain halts,

$$\Omega = \sum_{s \in \{0,1\}} P(s) = \sum_{U(p) halts} 2^{-|p|}$$

is random.

Simliarly, probabilities of other computer behaviours called Ω numbers (Becher,Chaitin 2001,2003; Becher,Grigorieff 2005,2009, Becher,Figueira,Grigorieff,Miller 2006; Barmpalias 2016)

Ω es aleatoria

Theorem

La expansión del número $\Omega = \sum_{U(p)\downarrow} 2^{-(|p|)}$ *es aleatorio.*

Demostración

Sea p el siguiente programa: $b_1 b_2 \ldots b_c \Omega[1..i]^*$. Consideremos $g : N \to \sum^*$ una enumeración de los programas que se detienen.

1. Leer $\Omega[1..i]^*$ y computar $\Omega[1..i]$.
2. m=1. Mientras $\left(\sum_{j=1}^{m} 2^{-|g(j)|} \leq \Omega[1..i]\right)$, $m = m + 1$;
3. Sea $O = \{U(g(j)) : 1 \leq j \leq m\}$
4. Sea s la cadena lexicograficamente menor tal que $s \notin O$.
5. Output s

Veamos que $|s^*| > i$. Para demostrarlo supongamos lo contrario, es decir, $|s^*| \leq i$. Entonces,

$$\Omega > \sum_{j=1}^{m} 2^{-|g(j)|} + 2^{-|s^*|}, \qquad \text{porque } \Omega \text{ tiene infinitos aportes}$$

$$> \Omega[1..i] + 2^{-|s^*|}) \qquad \text{porque } \sum_{j=1}^{m} 2^{-|g(j)|} > \Omega[1..i]$$

$$\geq \Omega[1..i] + 2^{-i} \qquad \text{porque supusimos } |s^*| \leq i$$

$$\geq \Omega \qquad \text{porque } \Omega[1..i] \text{ contiene exactamente los primeros } i \text{ bits de } \Omega.$$

Llegamos a que $\Omega > \Omega$, y esto es imposible.
Por lo tanto, $|s^*| > i$. Luego,

$$i < |s^*| = K(s) \leq c + |\Omega[1..i]^*| = c + K(\Omega[1..i])$$

Concluímos que para todo i,

$$K(\Omega[1..i] > i - c$$

□