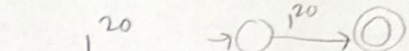
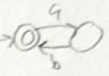
Kolmogorov complexity is, roughly speaking, a measure of how compressible objects are. More formally, the Kolmogorov complexity of an object is the length of its shortest description.

Ex) - ||||||||| |||| |||| |||| → 20 ones, 1^{20}, 

- $\{\varepsilon, ab, abab, ababab, \ldots\}$ → The set as written, 

, $(ab)^*$

- The classroom itself → Noyce 3820

- Chairs → a chair, the chair, all chairs
(chair suffices to describe a particular arrangement of the universe and/or something user as such)

This definition is still vague. How do we describe something? Why that way and not another?

It turns out it doesn't matter!

Thm (Informal Invariance Thm)

Given any descriptive language L, the optimal description language is at least efficient as L barring some constant overhead.

(Pf) Let x be some object and $D_x \in L$ its description in terms of L. We can write a finite algorithm $|A|$ to transform things in L to any other descriptive language L' (including the optimal one for x). But then $A(D_x) \in L'$ is a description of x of length

$$|D'_x| = |D_x| + |A|.$$

Since $|A| < \infty$ and works regardless of x, we're done.

□

Now let's formalize what we mean by KC and the Invariance Thm.

For a Turing complete descriptive language L (something that suffices to describe everything computable), we write the length of the shortest description of an object x as $K_L(x)$.

Thm (Invariant Thm)

If K_{L_1} and K_{L_2} are the complexity functions for (Turing complete) descriptive languages L_1 and L_2, then $\exists\, c>0$ (which depends only on L_1 and L_2) such that for every object x,

$$|K_{L_1}(s) - K_{L_2}(s)| \leq c.$$

Pf) By symmetry, it suffices to show $K_{L_2}(s) \leq K_{L_1}(s) + c$.
Since L_1 is Turing complete, there is an intepretor $I \in L_1$ which is a program that describes things in L_2 in terms of L_1.
Clearly, $K_{L_1}(I) < \infty$ and is constant, so if we pick $c = K_{L_1}(I)$,

$$K_{L_2}(x) \leq K_{L_1}(x) + K_{L_1}(I) = K_{L_1}(x) + c.$$

□

A theorem that we'll learn later is that there is a universal machine U that can simulate all other machines on any input. The manner in which we describe programs to it (and inputs) is a descriptive language! We then can say the complexity of an object x (usually simply referred to as a string) is the length of the shortest program which U reads in and, when simulated, outputs x and halts.

Generally, we write $K_U = K$.

From now on, we'll just refer to objects as strings.

There are a few basic theorems of note.

Thm) There is a constant $c > 0$ such that for every string s, $K(s) \leq |s| + c$.

Pf) Given s, we can write a program P_s that simply outputs s using a full description of s (eg. $P_{101101} = \{$ Return "101101" $\}$). P_s is a fixed amount bigger than s regardless of s, so we're done.

In other words, no string's complexity is too much bigger than itself. However, there are strings with arbitrarily large complexity.

Thm) $\forall n \in \mathbb{N}\ \exists s : K(s) \geq n$.

Pf) If not, then infinitely many strings could be described with finitely many programs of some maximum length. This is absurd since no function can be surjective if its domain is smaller than its codomain. □

Now here's the central trouble of K.

Thm) K is not computable.

Pf) For the sake of understanding (and elegance), we will treat this with some informality.
Suppose K is computable. Then by the well ordering principle (we can enumerate all strings and thus order them like the naturals), there is a "least string s that cannot be described by less than 20 English words." But we just described s with 13 (14 if we count the absent article) words. This is a contradiction, thus K is not computable. □

Does this then make K and Kolmogorov complexity useless? No! Even though we can't compute much of it, we can still prove useful facts. It also illuminates a deeper truth. Most things aren't computable!

Thm Let $N \subseteq \mathbb{N}$ with $|N| = |\mathbb{N}|$. Then $\exists n \in N : K(n) \notin O(1)$.

Pf There are only $O(1)$ descriptions of $O(1)$ length, but N is countably infinite, hence $\exists n \in N : K(n) \notin O(1)$. □

So far we've been working with the unconditional Kolmogorov complexity $K(s)$. We can define the conditional version as well. $K(s)$ uses a program p so that $U(p) = s$, and we minimize $|p|$. $K(s|y)$ instead uses programs p that get y for free, that is $U(p, y) = s$ and we still minimize only $|p|$. Note that in either case, if no such p exists, we say the complexity is ∞. You might notice that $K(s) = K(s|\varepsilon)$.

So what has been the point of all this? Well, we can characterize the regular languages as (roughly speaking) those for which, given n for free, has a constant-sized program which can output it's n^{th} string (in lexicographical order).

Define the following. Let Σ be a nonempty finite alphabet. Let y_i be the i^{th} element of Σ^* in lexicographical order for $i \geq 1$ (0 1 00 01 10 11 000 $\cdots$ for $\Sigma = \mathbb{Z}_2$). For $L \subseteq \Sigma^*$ and $x \in \Sigma^*$, let $\chi_x = \chi_{x,1}\chi_{x,2}\chi_{x,3}\cdots$ be the characteristic sequence of $L_x = \{y \mid xy \in L\}$, where

$$\chi_{x,i} = \begin{cases} 1 & xy_i \in L \\ 0 & xy_i \notin L. \end{cases}$$

We denote $\chi_{x,1}\chi_{x,2}\cdots\chi_{x,n} = \chi_x[1:n]$.

We now present without proof a characterization of the regular languages.

Thm Let $L \subseteq \Sigma^*$. Then the following statements are equivalent.

i) L is regular.

ii) $\exists c_L > 0 \; \forall x \in \Sigma^* \; \forall n \geq 1, \; K(\chi_x[1{:}n] \mid n) \leq c_L$.

iii) $\exists c_L > 0 \; \forall x \in \Sigma^* \; \forall n \geq 1, \; K(\chi_x[1{:}n]) \leq K(n) + c_L$.

iv) $\exists c_L > 0 \; \forall x \in \Sigma^* \; \forall n \geq 1, \; K(\chi_x[1{:}n]) \leq \log n + c_L$.

Here's how we use complexity (without the theorem for the moment), which will demonstrate logic that leads to a corollary of the Thm above.

Ex) Prove $L = \{1^p \mid p \text{ prime}\}$ is irregular. $(\Sigma = \{1\})$

Let p_{k+1} be the $k+1^{\text{th}}$ prime and p_k the k^{th} prime.

Consider the string $x_k = 1^{p_k}$. Then the first $y_k \in L_{x_k}$ is $y_k = 1^{p_{k+1} - p_k}$. However, the difference between primes grows unbounded. This means that there must be infinitely many such differences. In turn, it follows that there are infinitely many χ_x's for L. To see why, let D be the set of these differences. Then for $m, m' \in D$ with $m \neq m'$, $\exists k, k'$ such that $y_k = 1^m$, $y_{k'} = 1^{m'}$, and $\chi_{x_k}[1{:}m] = 0^{m-1}1$ and $\chi_{x_{k'}}[1{:}m'] = 0^{m'-1}1$. Since $m \neq m'$, these must disagree at bit m or bit m'.

But then $K(\chi_{x_k}[1{:}n] \mid n) \geq K(k)$ since we need k to specify x_k to specify χ.

Pick $k \notin O(1)$ with must exist since $k \in \mathbb{N}$. □

From this example, we can specify a useful corollary of the theorem. We levered the fact that there are infinitely many χ's when L is regular. This is not true.

Cor Let $L \subseteq \Sigma^*$ be regular, and let $\phi : \mathbb{N} \to \Sigma^*$ enumerate Σ^*. Then $\exists c > 0$ depending only on L and ϕ such that $\forall x \in \Sigma^*$, if y is the n^{th} string enumerated in L_x (or $\overline{L_x}$), then $K(y) \leq K(n) + c$.

The trick is not to pick ϕ which encodes y but pick x which encodes a complexy.

This follows from ϕ and the DFA for L (and thus L_x) being constant size wrt y. We need only specify n and check if the DFA accepts $\phi(n)$.

Ex) Using the corollary, prove $\{xx^R \mid x \in \Sigma^*\}$ is irregular.

Assume L is regular. ϕ is lexicographic.

Pick $x = (01)^m$ with $K(m) \notin O(1)$. Then $y = (10)^m$ is the first (and only) element of L_x, so $K(y) \leq K(1) + O(1) \Rightarrow K(y) \in O(1)$.

But clearly $K(y) \geq K(m) \notin O(1)$.

$\rightarrow\leftarrow$

$\therefore$ L is not regular. □

Ex) Using the corollary, prove $\{w \in \Sigma_2^* \mid \#(w,0) > \#(w,1)\}$.

Assume L is regular. ϕ is lexicographic.

Pick $x = 1^m$ with $K(m) \notin O(1)$. Then $y = 1^{m+1}$ is the first element of L_x, so $K(y) \leq K(1) + O(1) \Rightarrow K(y) \in O(1)$.

But $K(y) \geq K(m) \notin O(1)$.

$\rightarrow\leftarrow$

$\therefore$ L is not regular. □

Ex) Using the theorem prove Σ^* is regular.

$\chi = 1^\infty$ regarless of choice of x, so clearly $K(\chi[1{:}n] \mid n) = K(1^n \mid n) \in O(1)$, so Σ^* is regular. □

Ex) Prove using the theorem that $E = \{w \in \Sigma^* \mid |w| \text{ even}\}$ is regular.

All of χ regardless of x can be easily generated just by parity checking, so $K(\chi[1{:}n] \mid n) \in O(1)$, thus L is regular. □

Another application of KC.

Ex) Prove comparison-based sorting has a worst case lower bound $\Omega(n \log n)$.

Given an input $A \in S_n$, it suffices to identify the required swaps to reach the identity element via comparisons.

To do so, you give a description of A via which elements to swap.

It must be the case that $\exists B \in S_n : K(B) \in \Omega(\log |S_n|)$
$= \Omega(\log n!) = \Omega(n \log n)$.

Any program P which describes B on some input A (in this case just B) via sorting A must then satisfy

$$K(P) \geq K(B) \in \Omega(n \log n).$$

But then to execute $P(B)$ requires $\Omega(n \log n)$ time at minimum to terminate, thus comparison-based sorting has a worst case lower bound of $\Omega(n \log n)$.

□

It's worth noting that the above argument applies to non comparison-based sorts as well, but the comparison of $n \log n$ to pseudo polynomial functions is messy, which such algorithms tend to have.