CIS 511: Theory of Computation **Feb 9, 2017**

Lecture 9: Finishing Decidability; Starting Complexity

Professor Sampath Kannan *Zach Schutzman*

NB: *These notes are from CIS511 at Penn. The course followed Michael Sipser's Introduction to the Theory of Computation (3ed) text.*

Finishing Undecidablility

Recall Rice's Theorem states that any non-trivial language property of Turing machines is undecidable. We showed this via mapping reduction from A_{TM}.

Let's look at one more undecidable language. We'll think about randomness. Consider a Turing machine M on input x. At the end of computation, the tape has y on it.

Definition 9.1 *The **descriptive** or **Kolmogorov-Chaitin complexity** of a string $\mathcal{K}(y)$ is the length of the shortest description $\langle M \rangle, w$ such that M on input w halts with y on its tape.*

How do we go about writing the description of a Turing machine? One simple solution would be to duplicate each symbol in the representation, then terminate with a non-duplicated pair. For example, 10110101 *becomes* 1100111100110011001110*, where* 10 *is the terminating pair. Alternatively, we can write the length of the machine with the doubling, finished with a non-duplicate pair.*

For any x, $\mathcal{K}(x) \leq |x| + c$, for some constant c. To see this, just let M be the machine that immediately halts. It has some length c', which is fixed and should be fairly small. Using our first naive encoding, $2c' + |x| + 2$ is sufficient to encode x.

Definition 9.2 *A string x is **incompressible by** c if $\mathcal{K}(x) \geq |x| - c$. That is, x has no description that is c shorter than itself.*

Definition 9.3 *A string is **incompressible** if $\mathcal{K}(x) \geq |x|$.*

Claim 9.4 *There exist incompressible strings of every length.*

Proof: *Note that there are 2^n strings of length n (on a binary alphabet). The number of shorter descriptions is $\sum_{i=0}^{n-1} 2^i = 2^n - 1$. There are strictly fewer descriptions than strings, so there must be some string not representable by a shorter length, i.e. incompressible.*

■

How many strings of length n are compressible by 2? We want a description of length at most $n - 3$. So $\sum_{i=0}^{n-3} = n^{n-2} - 1$. That is, at least three quarters of strings of any length are incompressible by 2.

If x, y are strings, and c a constant:

- $\mathcal{K}(xx) = \mathcal{K}(x) + c$
- $\mathcal{K}(xy) \leq \mathcal{K}(x) + \mathcal{K}(y) + 2\log_2(\mathcal{K}(x)) + c$, *because we need to encode a length for* x *at the beginning.*

Let $L = \{x | x$ *is incompressible by* $2\}$. *Is* L *decidable?*

Claim 9.5 *No.*

Proof:

Assume, for the sake of contradiction, that L *is decidable, and has a decider* D. D *has description length* c. *Define a machine* M, *which generates each string of length* N, *then calls* D *on that string.* M *halts at the first string which* D *says is incompressible.* M *needs* N *as input, which has length* $\log_2(N)$, *plus it has* D*'s description, of length* c, *plus some other constant piece to describe* M *with length* c'. *So we have input of string* $2(c + c' + \log(N))$ *which* D *says is incompressible. But this string of length* $2(c + c' + \log(N))$ *is shorter, therefore the string is compressible, which is a contradiction.*

■

Time Complexity

Definition 9.6 *A* ***step*** *of a Turing machine is one execution of a transition.*

Suppose M *is a deterministic TM to decide* L.

Definition 9.7 *We say* M ***runs in time*** $f(n)$ *if for any input of length* n, M *terminates in at most* $f(n)$ *steps.*

Definition 9.8 *The complexity class* ***DTIME(****$f(n)$****)*** *is the set of all languages* L *for which a deterministic Turing machine takes* $O(f(n))$ *steps and decides* L.

Example: *DTIME(n)*$\subset$ *DTIME(*n^2*)*

Example: *Let* $L = \{0^k 1^k | k \geq 0\}$. *A TM to decide this can do our usual matching thing, where we move back and forth, matching characters. This requires* $O(n^2)$ *steps. We therefore have* $L \in$*DTIME(*n^2*). Can we do better? Yes. Cross off every other* 0, *then every other* 1, *and at each pass, check the parity of the number remaining characters. If this is ever odd, reject. If we ever cross out all of one character and have some of the others, reject. Otherwise, accept. To do this, we do* $\log(n)$ *rounds of crossing out, and we need 2 passes of* n *steps to do each round, for complexity of* $n\log(n)$. *We can therefore say that* $L \in$*DTIME(*$n\log n$*).*

It turns out we can't do better than this. Any language that a single-tape deterministic TM decides in time $o(n\log n)$ *is regular. We know* L *is not regular, so we can't do better than* $O(n\log n)$.