CSE 531: Computational Complexity I **Winter 2016**

Lecture 1: Course Overview and Turing machine complexity

January 6, 2016

Lecturer: Paul Beame *Scribe: Paul Beame*

1 Course Outline

1. Basic properties of Turing Machines (TMs), Circuits & Complexity
2. P, NP, NP-completeness, Cook-Levin Theorem.
3. Hierarchy theorems, Circuit lower bounds.
4. Space complexity: PSPACE, PSPACE-completeness, L, NL, Closure properties
5. Polynomial-time hierarchy
6. #P and counting problems
7. Randomized complexity
8. Circuit lower bounds
9. Interactive proofs & PCP Theorem - Hardness of approximation
10. Communication complexity (?)
11. Strong Exponential Time Hypothesis (SETH). COnnecting the complexity of SAT and polynomial-time problems.

Logistics The course website is http://www.cs.washington.edu/531. Coursework:

- problem sets (3 or 4, 50% total)
- take-home midterm (15-20%)
- in-class exam (30-35%)
- extra credit homework problems and class participation

2 Uniform vs non-uniform models of computation

Uniform models of computation - one device for all input sizes: Turing machines

Non-uniform models of computation - one device for each input size: e.g. Circuits.

We will look at the complexity of functions $f : D \to R$. We will focus on Boolean functions; i.e., $D = \{0,1\}^*$ and $R = \{0,1\}^*$ or $\{0,1\}$.

Multitape Turing Machines A (multitape) Turing Machine (TM) M consists of the following:

- finite set of states Q
- a state register
- storage: tapes divided into cells, each of which can store an a symbol from the alphabet Γ.
- heads that scan one cell of each tape.
- input alphabet $\Sigma \subset \Gamma$, $\{\square(\text{blank}), > (\text{start})\} \subseteq \Gamma \setminus \Sigma$. All but a finite portion of each tape contents consist of the blank symbol.
- k read/write work tapes 1-way infinite
- read-only input tape, initially $> x_1 \dots x_n$.
- write-only output tape; unnecessary if output is in $\{0,1\}$ - use special state q_{accept} to indicate output 1 in that case.
- transition function
$$\delta : Q \times \Gamma^{k+1} \to Q \times \Gamma^{k+1} \times \{L, S, R\}^{k+1}.$$
The $k+1$ symbols in the domain (being read) are from the input tape and k work tapes; the $k+1$ symbols in the co-domain (being written) are from the k work tapes and the output tape.

The Turing Machine reads one symbol from each tape based on the position of each head. Each head moves left, right, or stays where it is depending on the value of the corresponding element of $\{L, S, R\}$. It runs for ever or until it reaches a halting state. We write $M(x)$ for the contents of the output tape of when x halts.

In the same journal issue in which the original paper where Turing described the TM appear, there was another paper by Post in which he defined a very simular computational model. (Excerpts of both papers are linked on the course website.) There are a number of reasons stemming from

that original paper that the model has become so closely linked to Turing. Turing gave persuasive arguments that his model was both natural and sufficient to cover every possible computation. He showed the existence of a universal Turing machine. Finally, he showed the undecidability of the halting problem as part of showing that there could be no model that both captures all possible computations and always halts.

There are many variations in the Turing machine model that are equivalent to each other. For example, one can easy simulate a 2-way infinite tape cells indexed by consecutive elemments of $\mathbb{Z}$ rather than a single 1-way infinite tape indexed by $\mathbb{N}$. In this simulation we can view the 2-way infinite tape as folded over itself, with a single leftmost cell representing a cell on the two-way infinite tape indexed by 0 and upper and lower tracks on the remaining cells representing the negativly and positivly-indexed cells on the two-way infinite tape, respectively.

Another variation is the *random access* or *index* TM, in which, instead of the read heads, there is a special tape register i that can be written by the TM and causes the value of x_i to be copied to a specific tape location whenever the TM enters of a special query state. Many of the results we will cover also apply to this model. None of these other deterministic variations will change the # of steps required to solve problems by more than a $polylog(|x|)$ factor.

Definition 2.1. *Let $f : \{0,1\}^* \to \{0,1\}^*$ and $T : \mathbb{N} \to \mathbb{N}$. M* computes *f* in time *T (usually written $T(n)$) iff $\forall n \in \mathbb{N}$ and for every input $x \in \{0,1\}^n$, M runs in at most $T(n)$ steps and halts with $f(x)$ on its output tape.*

Definition 2.2. *We define the complexity class of Boolean functions,*

$$\mathsf{DTIME(T(n))} = \{\mathsf{f} : \{0,1\}^* \to \{0,1\} \mid \textit{there is a TM } M \textit{ that computes } f \textit{ in time } O(T(n))\}.$$

We also interpret $\mathsf{DTIME(T(n))}$ *as a set of* languages $L \subseteq \{0,1\}^*$ *where* $L = \{x \mid f(x) = 1\}$.

Definition 2.3. *Let $f : \{0,1\}^* \to \{0,1\}^*$ and $S : \mathbb{N} \to \mathbb{N}$. M* computes *f* in space *S (usually written $S(n)$) iff $\forall n \in \mathbb{N}$ and for every input $x \in \{0,1\}^n$, M visits at most $S(n)$ cells on each work tape and halts with $f(x)$ on its output tape.*

Definition 2.4. *We define the complexity class of Boolean functions,*

$$\mathsf{DSPACE(S(n))} = \{\mathsf{f} : \{0,1\}^* \to \{0,1\} \mid \textit{there is a TM } M \textit{ that computes } f \textit{ in space } O(S(n))\}.$$

We also interpret $\mathsf{DSPACE(S(n))}$ *as a set of* languages $L \subseteq \{0,1\}^*$ *where* $L = \{x \mid f(x) = 1\}$.

The definitions of time and space on multitape Turing machines and the view that these are the right ways to capture computational complexity is due to Hartmanis and Stearns, and Hartmanis, Lewis, and Stearns in the early to mid 1960's.

Definition 2.5. $\mathsf{P} = \bigcup_{\mathsf{k}} \mathsf{DTIME(n^k)}$.

We begin by focussing on time complexity. It turns out that there are some time bounds T that do not correspond to running times of Turing machines. All of the time bounds that we consider are particularly well-behaved in the following sense.

Definition 2.6. *A function $T : \mathbb{N} \to \mathbb{N}$ is* time constructible *iff $T(n) \geq n$ and there is some TM M_T such that M_T computes the map*

$$x \mapsto [T(|x|)]$$

in time at most $T(n)$ where $[N]$ denotes the binary representation of N.

The functions n, $n \log n$, n^c, and 2^n are all time-constructible as are all the functions we will consider.

The Universal Turing Machine A *universal Turing Machine* is a Turing machine with two input arguments $x \in \{0,1\}^*$ and the binary encooding $[M]$ of a Turing machine M such that U halts iff M halts on input x and, if it does halt, $U(x, [M]) = M(x)$.

A universal TM may have a smaller tape alphabet, smaller state set, and smaller number of tapes than the machine M given by $[M]$. It will represent the elements of Γ in binary as $[a]$ for $a \in \Gamma$. It will have to use a tape to store a register holding an encoding $[q]$ of the current state $q \in Q$ of M, and (possibly) to store encodings of the $k+1$ symbols currently scanned by the heads of M. The contents of the k work tapes of M and the positions of their read/write heads are simulated on a fixed # of tapes of M. The general idea of the simulation is for the simulating machine to match up the scanned symbols with the transition table δ encoded in $[M]$ and simulate the actions of each step of M.

Definition 2.7. *A Turing machine M is* oblivious *the location of the read heads of M depend only on*

- *the length $|x|$ of the input x, and*
- *the time step t.*

Theorem 2.8. *There is an oblivious (universal) Turing machine U such that for all TMs M there is a $C > 0$ such that for all $x \in \{0,1\}^*$ and all time bounds $T(n) \geq n$, if M halts on input x within $T(|x|)$ steps then $U(x, [M]) = M(x)$ and U on input $(x, [M])$ halts in at most $CT(|x|) \log_2 T(|x|)$ steps.*

Proof sketch. We first describe a simpler version that proves this with time bound $CT^2(|x|)$ and only uses one work tape. The work tape of U will have $2k$ "tracks". There will be blocks of cells that represent elements of $(\Gamma \times \{\square, \checkmark\})^k$. The $\checkmark$ represents the position of the read head on the corresponding tape. The algorithm proceeds by
(1) sweeping across the occupied region of the work tape left to right to find the symbols currently

scanned, and hence the moves to execute using $[M]$,
(2) sweeping right to left to execute those moves for which the read/write head moves left or stays in place,
(3) sweeping left to right to execute the rightward head moves of M.
(4) moving back to the left end of the work tape.
The number of occupied cells on the work tapes of M is at most $T(|x|)$ so each step costs at most $O(T(|x|)$ steps; i.e. $O(T^2(|x|))$ total. The algorithm is essentially oblivious - to do so one needs to make sure that the length of the region being swept over in simulating time step t does not depend on the input (using $\max t, n$ works).

In order to improve the running time we first describe a simple alternative approach to simulating the tapes of $[M]$. In this version, we use k tracks that each use the simulation of a 2-way infinite tape and align the positions of the work tapes at the 0 position (left end) of the tapes. This way, there is no need to mark the head positions and no need to record the currently scanned positions separately. However, whenever a head position on a simulated tape moves to the right, the entire contents of that tape need to be shifted one simulated cell to the left, and whenever it moves to the left, the contents must be shifted to the right. It costs $O(T(|x|)$ steps of U to do the shift to simulate each step of M so again the total running time is $O(T^2(|x|))$.

To get the improved time bound, the simulating universal machine will need an extra work tape. The difficulty in improving this simulation is that each shift has to move the entire non-blank portion of the tape of M. The idea, due to Hennie and Stearns, is to have a more relaxed representation of each tape by using a dummy character $\not d$ that is ignored so that 0110, $\not d0\ \not d\ \not d11\ \not d0$, and $01\ \not d1\ \not d\ \not d0\ \not d$ all represent the same tape contents. The general idea is to periodically spread out the contents of the tape to have sufficient room (in the form of $\not d$ symbols) so that left and right shifts only need to move a small portion of the tape contents at each step.

More precisely, in addition to the leftmost cell, the two-way infinite simulation of a single tape will have consecutive zones $Z_0, Z_1, \ldots$, where Z_i is 2^i cells wide and represents the contents in L_i and R_i both left and right of the read head. The invariant that will be maintained is that 2^i of the 2^{i+1} values in $L_i \cup R_i$ will contain $\not d$. The balance between L_i and R_i will change over time, and after 2^i simulated steps it is possible that one of them will be packed so that there is no room to shift further. Every 2^i steps the algorithm will spread the contents out so that each of $L_0, R_0, \ldots, L_i, R_i$ is half filled with $\not d$ symbols. This requires shifting $\Theta(2^i)$ elements up to $\Theta(2^i)$ positions. If there were only a single tape this would cost the product of these two, which would lead only to an $O(T^2(n))$ running time. However, with the second tape these shifts can be done in $O(2^i)$ time by copying the contents using the second tape. Since the algorithm runs only for $T(n)$ steps and $T(n) \geq n$, only $\log_2 T(n)$ zones are required. Therefore the total cost is

$$\sum_{i=0}^{\log_2 T(n)} c2^i \frac{T(n)}{2^i} = O(T(n) \log T(n)).$$

The textbook contains more details (but neglects to mention the need for the second tape).

In all of the above it will be important for some of the applications that U is not only oblivious but each head position is also very simple and easy to compute as a function of $|x|$, $|[M]|$, and t. $\square$