6.841 Advanced Complexity Theory Mar 5, 2007

Lecture 8

Lecturer: Madhu Sudan *Scribe: Hooyoung Chung*

1 Overview

- Alternation
- Fortnow's Theorem: $SAT \notin \mathrm{L}$ *or* $SAT \notin \bigcap_{\varepsilon>0} \mathrm{TIME}(n^{1+\varepsilon})$

2 Aside: Definition of $\mathrm{TIME}(t(n))$

Consider $PALINDROME = \{x \cdot x^{\mathcal{R}} \mid x \in \Sigma^*\}$. Is $PALINDROME \in \mathrm{TIME}(O(n))$? If we can use two tapes, the answer is clearly *yes*. On the other hand, if our measure of time complexity involves one-tape TMs, then it can be shown that *PALINDROME* requires $\Omega(n^2)$. This bound conflicts with our intuition, which says that *PALINDROME* is a simple problem with an obvious linear time solution. We conclude that one-tape TMs are a poor model of computation where time is concerned.

How about two-tape TMs? It can be shown that, potentially at the cost of extra space, a two-tape TM can simulate the computation of a generic multitape TM with only logarithmic slowdown. As a matter of convention, $\mathrm{TIME}(t(n))$ will generally denote the class of problems decidable by machines with any constant number of tapes in time $\leq t(n)$).

An "extra credit exercise." Build a universal one-tape TM U that, on input $\langle M, x\rangle$, simulates M on x with sublogarithmic slowdown, where M is a one-tape TM; that is, if M runs in $t(n)$, U should output $M(x)$ in $o(t(n) \lg t(n))$.

3 The Alternating TM

The alternating TM is a generalization of the nondeterministic Turing machine, in that it is allowed to perform many "threads" of computation simultaneously. We augment the deterministic TM model with two special kinds of states: an *existential* ($\exists$) and a *universal* ($\forall$), which are allowed to transition to two other states at once. We introduce the concept of states accepting or rejecting, where this behavior is governed by the type of state. $\exists$ states accept when at least one outgoing transition accepts; $\forall$ states accept when both outgoing transitions accept. A normal state has exactly one outgoing transition, and accepts when that transition accepts. The alternating TM accepts when its start state accepts.

The computation of an alternating TM M on an input w is naturally represented as a *computation tree*, whose nodes are the configurations M enters when it is run on w. A configuration incorporates the tape contents, head position, and finite state of M. The tree is rooted at the start configuration, and we draw an edge from configuration a down to b if a transition takes a to b.

When we discuss alternating TMs, we are interested in these resources:

- *Time*, measured as the depth of the computation tree. If an alternating TM M has computation tree depth $\leq t(n)$ on all inputs of size n, then $L(M) \in \mathrm{ATIME}(t(n))$.
- *Space*, the maximum space used in any configuration in the computation tree. If an alternating TM M terminates on all inputs of size n and uses space $\leq s(n)$ along any path, then $L(M) \in \mathrm{ASPACE}(s(n))$.
- *Alternation* counts the maximum number of times the machine alternates between existential and universal states and vice versa, along any path in the computation tree (normal states are not considered). For instance, any deterministic TM is an alternating TM with zero alternations.

Motivation for studying alternation

Why study alternation?

1. It models interesting problems; for instance, the set of problems decidable in polynomial time using only existential states is exactly NP, problems decidable in polynomial time using only universal states = coNP. An example of a problem which (we think) requires $\exists$ followed by $\forall$ is formula minimization, i.e.,
$$\{(\phi, k) \mid \exists \psi\colon |\psi| \leq k \text{ and } \forall x\colon \phi(x) = \psi(x)\}$$
2. It offers useful perspective for the comparison of time and space, as shown by the containments:
$$\text{NSPACE}(s(n)) \subseteq \text{ATIME}((s(n))^2) \subseteq \text{SPACE}((s(n))^2)$$
$$\text{TIME}(2^{s(n)}) \subseteq \text{ASPACE}(s(n)) \subseteq \text{TIME}(2^{O(s(n))})$$

4 Fortnow's Theorem

Fortnow's theorem states that $SAT \notin \text{L}$ or $SAT \notin \text{TIME}(n^{1+\varepsilon})$ for some $\varepsilon > 0$. We believe P $\neq$ NP, which implies both of these trivially, but have not been able to prove it.

The conventional statement of Cook's theorem is that all nondeterministic-polynomial-time-computable languages have a polynomial time reduction to SAT. In our proof, we will use the following stronger version, which gives useful upper bounds on the running time and space of the (optimal) reduction.

Theorem 1 (Cook) *Given an increasing function $t(n)$, there is a c such that for any nondeterministic TM M mapping reduction from $L(M)$ to SAT running in time $t(n)$, there is a mapping reduction f_M from $L(M)$ to SAT which is computable in time $O(t(n) \lg t(n))$ and space $c \lg t(n)$ and yields a formula of size $O(t(n) \lg t(n))$.*

(For a proof, see S. Cook, Short propositional formulae represent non-deterministic computation, *Information Processing Letters*, 26:269-270, 1988.)

The heart of the proof of Fortnow's theorem is in the following lemmas.

Claim 2 $SAT \in \text{L} \implies \text{NTIME}(t(n)) \subseteq \text{SPACE}(c \lg t(n))$ *for some constant c. (The idea: if $SAT \in$ L, "all computation is small space.")*

Proof From theorem 1, we can reduce any problem in NTIME($t(n)$) to SAT using space within some factor of $\lg n$. If $SAT \in$ L, then $SAT \in \text{SPACE}(c_0 \lg t(n))$ for some c_0, so we can solve any problem in NTIME($t(n)$) in space $c \lg t(n)$ for some c. ■

Claim 3 *For any $\varepsilon > 0$, $SAT \in \text{TIME}(n^{1+\varepsilon}) \implies \text{ATIME}(a, a \cdot t(n)^{c/a}) \subseteq \text{ATIME}(a-1, a \cdot t(n)^{(c/a)(1+\varepsilon)})$. (If $SAT \in \text{TIME}(n^{1+\varepsilon})$, "alternation is not powerful.")*

Claim 4 $\text{SPACE}(s(n)) \subseteq \text{ATIME}(a, a \cdot 2^{s(n)/a})$. *("Alternation is powerful (for small space computation).")*

Proof Idea Proof similar to the standard proof of Savitch's theorem. Given a deterministic TM M running in time $t(n)$, we can find out if it accepts a string by looking for a path of length $t(n)$ between its start and accepting configuration on that string. The way to do this is to guess the configuration in the middle and recursively check if paths of length $t(n)/2$ exist from the start to the midpoint and from the midpoint to the end. A deterministic TM which uses space $s(n)$ uses time $2^{s(n)}$. ■

Notation for alternating time. Note the use of $\mathrm{ATIME}(a, t(n))$; this denotes the classs of languages decidable in $t(n)$ by alternating TM that make a constant $\leq a$ alternations, e.g., $\mathrm{ATIME}(0, t(n)) = \mathrm{TIME}(t(n))$. (Previously we defined $\mathrm{ATIME}(t(n))$, the class of languages decidable in time $t(n)$ for some alternating TM using an unrestricted amount of alternation.)

Theorem 5 (Fortnow) *For any $c < \infty$, we can find $\varepsilon > 0$ such that*

$$SAT \in \mathrm{SPACE}(c \lg n) \implies SAT \notin \mathrm{TIME}(n^{1+\varepsilon}).$$

(This statement of Fortnow's theorem is equivalent to the one in the overview.)
Proof Assume that $SAT \in \mathrm{L} \cup \mathrm{TIME}(n^{1+\varepsilon})$.

$$\begin{aligned}
\mathrm{TIME}(t(n)) &\subseteq \mathrm{NTIME}(t(n)) \\
&\subseteq \mathrm{SPACE}(c \lg t(n)) && \text{by Claim 2} \\
&\subseteq \mathrm{ATIME}(a, a \cdot t(n)^{c/a}) && \text{by Claim 4} \\
&\subseteq \mathrm{ATIME}(a-1, a \cdot t(n)^{(c/a)(1+\varepsilon)}) && \text{by Claim 3} \\
&\subseteq \mathrm{ATIME}(a-2, a \cdot t(n)^{(c/a)(1+\varepsilon)^2}) && \text{likewise} \\
&\quad \vdots \\
&\subseteq \mathrm{ATIME}(0, a \cdot t(n)^{(c/a)(1+\varepsilon)^a}) \\
&= \mathrm{TIME}(a \cdot t(n)^{(c/a)(1+\varepsilon)^a})
\end{aligned}$$

which can only be true if $\frac{2c}{a}(1+\varepsilon)^a \geq 1$. ■