CSC 2401, Introduction to computational complexity
Lecture #22, Tuesday, November 27, 2001
Lecture: Allan Borodin
Scribe: Yubin Huang

1 EXP-Complete Problem

1.1 Totality of Extended Regular Expression

Definition: We say a string is *an extended regular expression* if it is a regular expression with an additional intersection operation $\bigcap$.

Totality of Extended Regular Expression (TERE): Given an extended regular expression R over set Σ, determine whether $L(R) = \Sigma^*$. Hence,

$\underline{TERE} = \{R|R$ is an extended regular expression: $L(R) = \Sigma^*\}$.

$\overline{TERE} = \{R|R$ is an extended regular expression: $L(R) \neq \Sigma^*\}$.

Theorem: TERE is EXPSPACE-complete.

Similarly with TRE is in PSPACE-complete, the idea of the proof is that we can encode computational paths whose configurations are exponential in n. The idea is to use a very stylized encoding of computations. We will use two "tracks" to encode a computation. A pair of symbols (one from each track) then defines a new alphabet Γ over which we construct our extended regular expressions. On one track (lets say the bottom track) we will be running a $mod 2^n$ counter; that is, the bottom track will look like the following: $\#\bar{0}\#\bar{1}\ldots\#\overline{2^n-1}\#\bar{0}\#\bar{1}\ldots$
where $\bar{i}$ is a binary encoding (with leading zeros) of i using exactly n bits. We will place the actual symbols of a configuration on the top track say on top of the # symbols on the lower track. This allows us to have configurations of length 2^n. In order to match up corresponding locations of consecutive configurations we need to use the intersection operation $\bigcap$. Omitting the details we get:

Let M be a non-deterministic 2^n space bounded Turing Machine. We claim that there is a transformation $x \vdash R_{m,x} = R_x \;\; : x \in L(M)$ if and only if $L(R_x) \neq \Gamma^*$.

1.2 Another Example

One more example of a natural provably hard language is the 1^{st} order theory of the reals with the binary operations $+$ and $*$ and the constants 0 and 1.

For example, we can write sentences of the form $\forall z \exists x \forall y : [x \cdot x + y = z \wedge \ldots]$ and asked whetehr any such sentence is true or false when interpreted over the reals. It is well known that the analogous $1`st$ order theory of the integers is undecideable.

But the 1^{st} order theory of the reals is decidable and any sentence (of say length n) can be decided within deterministic time $2^{2^{O(n)}}$. Tarski first showed that this theory was decideable and later Collins and then Monk gave the double exponential time upper bound.

We state without proof that there is a non-deterministic or analogous "Σ_k") single exponential lower bound fort this decision problem. (In fact, the lower bound applies to Σ_k (exponential time) for any k.)

To prove a non deterministic lower bound (rather than a deterministic lower bound) we need the following non-trivial result. One proves the lower bound by showing how sentecnes in this theory can encode express the fact that a real number is an integer within some (doubly exponential) range. Integers of doubly exponential size can represent strings (and therefor computations) of exponential length.

Hierarchy Theorem: Let $t_1(n)$ and $t_2(n)$ be time constructible functions. If $t_1(n+1) = o(t_2(n))$, then $NTIME(t_2) \subset NTIME(t_1)$.

2 Time-Space Tradeoff

Definition: Let $t(n) \geq n$ be a time constructible function, $s(n) \geq \log n$ be space constructible function. $L = L(M) \in DTISP(t(n), s(n))$ for some deterministic Turing Machines running simultaneously in time $t(n)$ and space $s(n)$. Similarly, $L = L(M) \in NTISP(t(n), s(n))$ for some non-deterministic Turing Machines running simultaneously in time $t(n)$ and space $s(n)$.

Question: Can we show that there exists a natural language L such that L is not in $DTISP(n \cdot (\log n)^{O(1)}, \log n)$. We can get an "easy" time space lower bound for the one input head sequential access Turing Machine model. Specifically for the palindromes $L = \{\omega\omega^R | \omega \in \{0,1\}^*\}$, it is easy to prove that $t(n) * s(n) = \Omega(n^2)$. But in this case, we can decide membership in L by a linear time, log space Turing machine having two input heads. (Here we are counting the location of the heads as part of the space bound.) Hwence we do not consider the 1 head time space result as an "intrinsic" lower bound. It is too model dependent.

We note that there do exist time space results for Turing Machine with multiple read heads. But even these results seem too model dependent. We would like such a result for a model allowing "random access" to the input.

A great "intrinsic" result would be such a time space lower bound for branching programs. We knew that sorting n integers (say in the range $[1, n^2]$) requires $t(n) * s(n) = \Omega(n^2)$ in a Boolean Branching Program model but no such result is known for a decision problem. After a 20 year development, Ajtai was able to finally show that $\exists$ a polytime computable L such that any branching program deciding L within space $s(n) = n^{1-\varepsilon}$ requires time $t(n) = \Omega(n \cdot \sqrt{\frac{\log n}{\log\log n}})$.

Fortnow produced a more impressive lower bound for a general uniform model (even allowing random access of the input). His result that $SAT \notin DTISP(n^{1+o(n)}, n^{1-\varepsilon})$.

Following Fortnow, Lipton-Viglas showed that $SAT \notin DTISP(n^{\sqrt{2}-\varepsilon}, n^{o(1)})$.

Fortnow-Viglas showed that $\exists d, SAT \notin DTISP(n^{\frac{\sqrt{5}+1}{2}}, n^d)$.

SAT is compuable in non-detyerministic linear time and log-space. If $SAT \in DTISP(t, s)$, then $NTIME(n) \subseteq DTISP((\log n)^r \cdot t(n \cdot (\log n)^r), \log n + s(n \cdot (\log n)^r))$.

3 What next

These time-space lower bounds are derived by so-called "indirect diagonalization".