ITCS:CCT09 : Computational Complexity Theory Mar 9, 2009

Lecture 3

Lecturer: Jayalal Sarma M.N. *Scribe: Xiaohui Bei*

Last time we introduced the Hierarchy Theorem which discuss the relationship among deterministic complexity classes, in both time and space aspects. We also introduced non-determinism, and proved Savitch's Theorem which relates space-bounded nondeterminism to space-bound determinism.

In today's lecture, first we continue our discussion about the space-bounded nondeterminism. We show that most of such classes are closed under complements. Then we give a short description of oracle turing machines and notion of relativisation, and talk about the limitation of such property, which gives us a sense that why the Pvs NPquestion is so hard to settle.

1 Space-bounded Nondeterminism

For any complexity class C, we let coC be the class of complements of sets in C, i.e. $coC = \{L|\overline{L} \in C\}$. Deterministic complexity classes are closed under complementation, while this is not always true for undeterministic cases. For example, the relation between NPand coNP is still a open question left to us. However, if we consider the nondeterministic space-bounded classes C, the following theorem shows that C and coC are actually same.

Theorem 1 (Immerman-Szlepcsenyi theorem) *NSPACE*$(s(n))$ = *co-NSPACE*$(s(n))$ *for all* $s(n) \geq \log n$.

Proof We need to show that for any $L \in$ NSPACE$(s(n))$, there exist a NTM that computes $\overline{L}$. Recall that to compute $\overline{L}$ is to check if all computation paths start from α_0 (the initial configuration) lead to rejection.

We define a predicate $reach(\alpha, \beta, k)$, here α and β are both configurations and k is an integer. $reach(\alpha, \beta, k)$ means that β is reachable from α in at most k steps, that is, that there exist a sequence $\alpha = \alpha_0, \alpha_1, \ldots, \alpha_{k-1}, \alpha_k = \beta$, where $\alpha_{i-1} \rightarrow \alpha_i$ or $\alpha_{i-1} = \alpha_i$ for each $i = 1, 2, \ldots, k$. First we show that there exist a NTM which can computes $reach(\alpha, \beta, k)$ in space $O(s(n) + \log k)$.

Claim 2 $reach(\alpha, \beta, k)$ *can be tested in NSPACE*$(s(n) + \log k)$.

Proof The NTM M does the following: Let $\gamma_0 = \alpha$. For each $i = 1, 2, \ldots, k-1$. M guesses a configuration γ_i and verifies that $\gamma_{i-1} \to \gamma_i$ or $\gamma_{i-1} = \gamma_i$. If the test fails at some step, M refect. Finally M verifies that $\gamma_{k-1} = \beta$ and accepts if it holds.

It's easy to see that M accepts (α, β, k) if and only if $\mathit{reach}(\alpha, \beta, k)$. Storing the current configuration γ_i costs $O(s(n))$ space, and the counter of i costs $O(\log k)$ space. So M uses space $O(s(n) + \log k)$. ■

Having this problem solved, we can now use it as a subroutine to solve other problems. The next task would be constructing another NTM M_1 which computes N_k, the number of configurations reachable from a given configuration α in no more than k steps, using space $O(s(n) + \log k)$.

We let M_1 computes $N_1, N_2, \ldots, N_k$ one by one. First we know $N_0 = 1$ (α is the only reachable configuration). Now assume we have N_{k-1} (induction) and want to compute N_k. In order to do this, let M_1 runs the following procedure:

```
$N_k \leftarrow 0$
**for all** configuration $\beta$ **do**
  $\tau_\beta \leftarrow$ **false**
  **for** $i = 1$ to $N_{k-1}$ **do**
    Guess a configuration $\delta_i$.
    Verify $\mathit{reach}(\alpha, \delta_i, k-1)$.
    Verify $\delta_i \to \beta$ or $\delta_i = \beta$.
    If both of these conditions hold, set $\tau_\beta$ to **true**.
  **end for**
  **if** $\tau_\beta =$ **true then**
    $N_k \leftarrow N_k + 1$
  **end if**
**end for**
```

It's easy to see that M_1 uses space $O(s(n) + \log k)$ and it indeed computes the number of reachable configurations from a given configuration in no more than k steps. And using this as a subroutine, we can actually compute the number N of all reachable configurations from a given configuration. The idea is simple, just find the smallest k such that $N_k = N_{k+1}$, which means no new configurations can be reached after k steps. And $N = N_k$ is just the answer we wanted. Let m be the maximum length of an accepting path on some input x. Then, $m = 2^{O(s(n))}$. So we know $k \leq 2^{O(s(n))}$. Which means this NTM uses space $O(s(n) + \log k) = O(s(n))$.

Finally, we will use M and M_1 to construct our final NTM M_2 for $\overline{L}$ as follows:

First M_2 calls M_1 to compute the number N of all reachable configurations from the initial configuration α_0. Then we guess N configurations $\gamma_1, \gamma_2, \ldots, \gamma_N$ one by one, checks that each of them is reachable from α_0 (using NTM M) and none of them is an accepting path.

If all of the above conditions are checked, M_2 accepts, otherwise M_2 rejects the computation path.

Following is the pseudocode of NTM M_2:

```
$N \leftarrow$ number of reachable configurations from the initial configuration.
$k \leftarrow$ maximum length of an accepting path.
{The above two values can be computed by simulating $M_1$}
$flag \leftarrow$ **false**
**for** $i = 1$ to $N$ **do**
  Guess $\delta_i$.
  Check $reach(\alpha_0, \gamma_i, k)$ (by NTM $M$). If it's not true, $M_2$ rejects.
  **if** $\gamma_i$ is an accepting configuration **then**
    $flag \leftarrow$ **true**
  **end if**
**end for**
**if** $flag =$ **false then**
  $M_2$ accepts
**else**
  $M_2$ rejects.
**end if**
```

We claim that M_2 computes $\overline{L}$. First it's easy to see that if $x \notin L$, which means all reachable configurations from α_0 are rejecting. So M_2 will guess all the reachable configurations γ_i and at least accept x. If $x \in L$, then there exist one reachable configuration which is accepted. So either M_2 will guess all the reachable configurations which contain this one, or it will guess at least one nonreachable configuration. In either cases the computation will lead to rejection.

Finally, it's not hard to show that M_2 also uses $O(s(n))$ spaces, and by the tape compression theorem, we know the theorem holds. ■

This result together with the ones proved in the last two lectures, give us a relationship picture about time/space-bouneded complexity classes, which is the best we know so far.

2 Relativization

Roughly speaking, relativization of a statement with respect to (a language) A means giving each machine involved in that statement the ability to access an A-oracle. If the statement still holds, we say that it holds relative to A. And we say that a statement relativizes if it holds relative to any language.

A very revealing fact is that many results we have in complexity relativize. The reason be-

$$\begin{tikzcd}
 & & & \mathsf{NP} \arrow[r] & \mathsf{PSPACE} \\
\mathsf{L} \arrow[r] \arrow[rrrru, bend left, "\neq"] & \mathsf{NL} \arrow[r] & \mathsf{P} \arrow[ru] \arrow[rd, "\neq"'] & & \\
 & & & \mathsf{EXP} &
\end{tikzcd}$$

Figure 1: Relation among the complexity classes

hind this is that most of the techniques we used so far all relativize, such as diagonalization, etc.

However, the relativization property also gives limitation to these techniques on proving some results. Let's take the most famous Pvs NPproblem as an example. One can prove the following result:

Theorem 3 *There exist oracles A and B, such that* $\mathsf{P}^A = \mathsf{NP}^A$ *and* $\mathsf{P}^B \neq \mathsf{NP}^B$.

In other words, neither of the statements $\mathsf{P} = \mathsf{NP}$ and $\mathsf{P} \neq \mathsf{NP}$ relativize. Which means that none of the techniques that relativize have enough power to solve the Pvs NPquestion. In order to settle this ultimate problem, we need new ideas.