Lecture 4: COMPLEMENT OF SPACE CLASSES

1 REVIEW

Simulation by Time: the technique here is basically counting the number of possible configurations. This appears trivial, and yet, it is the best we know!

Simulation by Space: the key technique here is Savitch's result. Consequence: $PSPACE = DSPACE(n^{O(1)}) = NSPACE(n^{O(1)})$.

Simulation by Reversal: here, we try to do as much work in one phase as we can.

Today we examine another set of techniques for "complementing space classes".

2 INTRODUCTION

Running Complexity Classes: where we count ALL computation paths.

In this section, our complexity functions $s(n)$ has the property that $s(n)$ is defined and finite for all $n \in \mathrm{N}$. Thus if a TM that runs in space $s(n)$, it uses a finite amount of space on all computation paths (but may not halt).

We begin with the obvious remark:

$$DTIME_r(t) = co\text{--}DTIME_r(t).$$

But what about $DSPACE_r(s)$? The trick for time does not work now!

3 COMPLEMENT OF DETERMINISTIC SPACE

Theorem 1

$$DSPACE_r(s) = co\text{--}DSPACE_r(s)$$

Thus the result hinges on the ability to guarantee halting, and follows immediately from:

Lemma 2 *If* M *is deterministic and runs in space s then there is a deterministic* N *that runs in space s such that* N *halts on all computation paths.*

4 COMPLEMENT OF NONDETERMINISTIC SPACE

This result solves a major open problem going back to Kuroda in 1964. It was simultaneously obtained by Immerman and Szelepcsényi in 1987.

Theorem 3 *For $s(n) \geq \log n$,*

$$NSPACE_r(s) = co\text{--}NSPACE_r(s)$$

Let us fix a TM M that runs in space $s(n)$ and fix an input x of length n. Our goal is to construct a TM N that runs in space $s(n)$ that will reject x iff M accepts x.

The critical idea in this proof is the concept of a census function. For the parameters h, m, let $REACH_h(m)$ denote the set of configurations reachable from the initial configuration $C_0(x)$ using space at most h and time at most m. Let $\alpha_h(m) = |REACH_h(m)|$. This $\alpha_h(.)$ function is called the *census function* and is the key to this simulation. Note that $\alpha_h(.)$ is a non-decreasing function. In the following we assume $h \geq \log n$.

Suppose we can compute this function. Let us see how we can use it to compute the following predicate $CHECK(C, h, m, \alpha_h(m))$. Note that the last argument to $CHECK$ is logically speaking redundant, but computationally, it is not necessarily redundant.

[Program for CHECK]

Using a Turing machine N which has x on its input tape, and $C, h, m, \alpha_h(m)$ on its work tapes, we can compute $CHECK$ non-deterministically in space $h + \log m$. This Turing machine may loop. However, at least one path will terminate with the correct answer (accept or reject), and no paths gives the wrong answer.

Next we show how to compute the census function. Given $\alpha_h(m-1)$, we show how to compute $\alpha_h(m)$ non-deterministically. Again