Lecture 4

The Immerman–Szelepcsényi Theorem

In 1987, Neil Immerman [65] and independently Róbert Szelepcsényi [119] showed that for space bounds $S(n) \geq \log n$, the nondeterministic space complexity class $NSPACE(S(n))$ is closed under complement. The case $S(n) = n$ gave an affirmative solution to a long-standing open problem of formal language theory: whether the complement of every context-sensitive language is context-sensitive.

Theorem 4.1 (Immerman–Szelepcsényi Theorem) *For* $S(n) \geq \log n$, $NSPACE(S(n)) =$ *co-*$NSPACE(S(n))$.

Proof. For simplicity we first prove the result for space-constructible $S(n)$. One can remove this condition in a way similar to the proof of Savitch's theorem (Theorem 2.7).

The proof is based on the following idea involving the concept of a *census function*. Suppose we have a finite set A of strings and a nondeterministic test for membership in A. Suppose further that we know in advance the size of the set A. Then there is a nondeterministic test for *nonmembership* in A: given y, successively guess $|A|$ distinct elements and verify that they are all in A and all different from y. If this test succeeds, then y cannot be in A.

Let M be a nondeterministic $S(n)$-space bounded Turing machine. We wish to build another such automaton N accepting the complement of

$L(M)$. Assume we have a standard encoding of configurations of M over a finite alphabet Δ, $|\Delta| = d$, such that every configuration on inputs of length n is represented as a string in $\Delta^{S(n)}$.

Assume without loss of generality that whenever M wishes to accept, it first erases its worktape, moves its heads all the way to the left, and enters a unique accept state. Thus there is a unique accept configuration $\mathtt{accept} \in \Delta^{S(n)}$ on inputs of length n. Let $\mathtt{start} \in \Delta^{S(n)}$ represent the start configuration on input x, $|x| = n$: in the start state, heads all the way to the left, worktape empty.

Because there are at most $d^{S(n)}$ configurations M can attain on input x, if x is accepted then there is an accepting computation path of length at most $d^{S(n)}$. Define A_m to be the set of configurations in $\Delta^{S(n)}$ that are reachable from the start configuration `start` in at most m steps; that is,

$$A_m \quad = \quad \{\alpha \in \Delta^{S(n)} \mid \mathtt{start} \xrightarrow{\leq m} \alpha\}.$$

Thus $A_0 = \{\mathtt{start}\}$ and

$$M \text{ accepts } x \quad \Leftrightarrow \quad \mathtt{accept} \in A_{d^{S(n)}}.$$

The machine N will start by laying off $S(n)$ space on its worktape. It will then proceed to compute the sizes $|A_0|, |A_1|, |A_2|, \ldots, |A_{d^{S(n)}}|$ inductively. First, $|A_0| = 1$. Now suppose $|A_m|$ has been computed and is written on a track of N's tape. Because $|A_m| \leq d^{S(n)}$, this takes up $S(n)$ space at most. To compute $|A_{m+1}|$, successively write down each $\beta \in \Delta^{S(n)}$ in lexicographical order; for each one, determine whether $\beta \in A_{m+1}$ (the algorithm for this is given below); if so, increment a counter by one. The final value of the counter is $|A_{m+1}|$. To test whether $\beta \in A_{m+1}$, nondeterministically guess the $|A_m|$ elements of A_m in lexicographic order, verify that each such α is in A_m by guessing the computation path $\mathtt{start} \xrightarrow{\leq m} \alpha$, and for each such α check whether $\alpha \xrightarrow{\leq 1} \beta$. If any such α yields β in one step, then $\beta \in A_{m+1}$; if no such α does, then $\beta \notin A_{m+1}$.

After $|A_{d^{S(n)}}|$ has been computed, in order to test $\mathtt{accept} \notin A_{d^{S(n)}}$ nondeterministically, guess the $|A_{d^{S(n)}}|$ elements of $A_{d^{S(n)}}$ in lexicographic order, verifying that each guessed α is in $A_{d^{S(n)}}$ by guessing the computation path $\mathtt{start} \xrightarrow{\leq d^{S(n)}} \alpha$, and verifying that each such α is different from `accept`.

The nondeterministic machine N thus accepts the complement of $L(M)$ and can easily be programmed to run in space $S(n)$.

To remove the constructibility condition, we do the entire computation above for successive values $S = 1, 2, 3, \ldots$ approximating the true space bound $S(n)$. In the course of the computation for S, we eventually see all configurations of length S reachable from the start configuration, and can check whether M ever tries to use more than S space. If so, we know that S is too small and can restart the computation with $S + 1$. $\square$