CS221: Computational Complexity Prof. Salil Vadhan

Lecture 15: **EXPSPACE**-Completeness

10/25 *Scribe: Ruggiero Cavallo*

Contents

1 Announcements

- Pick up PS3.
- The material from this lecture is not covered in Papadimitriou.

2 EXPSPACE-Completeness

Regular expressions with exponentiation (r.e.e.'s) over Σ, $\emptyset$, $\cdot$, $\cup$, $\star$, $\uparrow$

- $a \in \Sigma$, $L(a) = \{a\}$
- $L(\emptyset) = \emptyset$
- $L(\varepsilon) = \{\varepsilon\}$, where ε is the empty string.
- $L(R_1 \cdot R_2) = \{xy : x \in L(R_1),\, y \in L(R_2)\}$ [concatenation]
- $L(R_1 \cup R_2) = L(R_1) \cup L(R_2)$
- $L(R^\star) = L(R)^\star = \{x_1 x_2 \cdots x_k : k \geq 0, \forall i\, x_i \in L(R)\}$
- $L(R \uparrow) = L(R \cdot R \ldots R)$ (note: succinctness of $\uparrow$ is tied to high complexity of the problem)

REE $= \{R : L(R) = \Sigma^\star\}$

Theorem 1 REE *is* **EXPSPACE**-*complete.*

Proof:

We'll prove the theorem in two parts. First, we'll show that REE is in **EXPSPACE**, and then that REE is **EXPSPACE**-hard.

- REE $\in$ **EXPSPACE**

Following is an **EXPSPACE** algorithm that decides REE. Through a series of conversions, we attain a TM that has a non-accepting state reachable from its initial state just in case the given regular expression doesn't generate $\Sigma^\star$.

Given a regular expression with exponentiation R of size n:

1. Convert R to a standard reg exp. (without exponentiation [$\uparrow$]) of size $2^{O(n)}$ (call it R').
2. Convert R' to an NFA (non-deterministic finite automaton) N of size $2^{O(n)}$.
3. Convert N to DFA M of size $2^{2^{O(n)}}$.
4. Test whether M has a non-accepting state reachable from its initial state.

But, we don't need doubly exponential space ($2^{2^{O(n)}}$). Why not? In Step 4, reachability on a graph of size $2^{2^{O(n)}}$ can be done in space $O\left(\log^2 2^{2^{O(n)}}\right) = 2^{O(n)}$ by Savitch's theorem. Step 3 can be viewed as a space $2^{O(n)}$ reduction. Instead of explicitly writing down its output M of size $2^{2^{O(n)}}$, whenever Step 4 needs to access part of M, we just rerun Step 3, using workspace $2^{O(n)}$.

- REE is EXPSPACE-hard.

Given L accepted by a space 2^{n^k} TM M, without loss of generality assume M has 1 tape.

Given input x, we want to construct a r.e.e. (regular expression with exponentiation) $R_{M,x}$ such that $L(R_{M,x}) = \Sigma^\star$ iff M doesn't accept x. (Note: proving that it *doesn't* accept x is OK, since **EXPSPACE** is closed under complement; we're actually reducing $\overline{L}$ to REE.)

$\Sigma = \{$tape symbols of $M\} \cup \{[q, x] : q$ is a state of M, x is a tape symbol$\} \cup \{\#\}$ (separator symbol)

Idea: A string $y \in \Sigma^\star$ will represent a computation of M on input x by including a sequence of configurations of M separated by #, e.g.,

$$\#\underbrace{abba[q_3, b]bba}_{\text{worktape: length exactly } 2^{n^k}}\#$$

(Note that we can pad the worktape with blank symbols to ensure the desired length.) From the above example, if the TM moves its head to the right and writes a, then we get:

$$\#abbaa[q_4, b]ba\#$$

We'll define $R_{M,x}$ such that

$$L(R_{M,x}) = \begin{cases} \Sigma^\star \setminus y & \text{if } y \text{ represents correct accepting computation of } M \text{ on } x \\ \Sigma^\star & \text{otherwise} \end{cases}$$

The intuition for the above is that incorrect computations can be captured by representing local inconsistencies. So, it accepts all but the correct computation. Once you've realized that this is what you want to do, the construction of $R_{M,x}$ is not too difficult.

The correct computation on $x_1 \cdots x_n$ is of the form:

$$\#[q_0,x_1]x_2\ldots x_n \underbrace{BB\ldots B}_{\substack{2^{n^k}-n \\ \text{blank} \\ \text{symbols}}} \#a[q_3,x_2]x_3\ldots x_n B\ldots B\#\ldots\#[q_{acc},B]B\ldots B\#$$

We can specify $R_{M,x}$ as follows:

$$R_{M,x} = \text{START} \cup \text{MOVE} \cup \text{FINISH}$$

where START represents computations that violate the first configuration, MOVE represents a violation of a transition of the TM, and FINISH reprsents expressions that don't contain an accept state. Here are the explicit definitions:

$$\text{FINISH} = (\Sigma - \{[q_{acc},x] : x \text{ tape symbol}\})^\star$$

Now, just like in the Cook-Levin construction, we know that a symbol of a TM configuration is determined by three adjacent symbols in the previous configuration (in the same position and the ones immediately adjacent). So, for three configuration symbols wxy, let $f(wxy)$ denote the symbol that should be in the position of x in the next configuration. Now we can define MOVE as follows:

$$\text{MOVE} = \bigcup_{wxy} \Sigma^\star wxy(\Sigma \uparrow 2^{n^k}-1) \underbrace{(\Sigma^3 \setminus \{f(wxy)\})}_{\substack{\text{triple of symbols} \\ \text{other than } w'x'y'}} \Sigma^\star$$

START should equal all strings not starting with $\#[q_0 x_1]x_2\ldots x_n \underbrace{BBB}_{2^{n^k}-n}\#$

$$\begin{aligned}\text{START} = \;& (\Sigma \setminus \{\#\})\Sigma^\star \;\cup \\ & \#(\Sigma \setminus \{[q_0,x_1]\})\Sigma^\star \;\cup \\ & \#[q_0,x_1](\Sigma \setminus \{x_2\})\Sigma^\star \;\cup\; \ldots \;\cup \\ & [\Sigma \uparrow (n+1)] \left[(\Sigma \cup \{\Sigma\}) \uparrow \left(2^{n^k}-n-1\right)\right] [\Sigma \setminus \{B\}]\,\Sigma^\star\end{aligned}$$

Thus formulated, $R_{M,x}$ generates $\Sigma^\star$ if and only if M doesn't accept x. It's now clear that any problem in **EXPSPACE** can be reduced to REE, and thus the proof is complete.

■

Things to look at when examining this proof:

- Why was $\uparrow$ important?
- Why was $\star$ important?
- Why look for non-accepting rather than accepting computations?

By the Space Hierarchy Theorem, we have:

Corollary 2 REE $\notin \bigcap_\epsilon \mathbf{SPACE}(2^{n^\epsilon})$

and since $\mathbf{TIME}(t) \subseteq \mathbf{SPACE}(t)$, we also have REE $\notin \bigcap_\epsilon \mathbf{TIME}(2^{n^\epsilon})$.

2.1 Commentary

- Actually, REE is complete for $\mathbf{SPACE}(2^{O(n)})$ under linear-time reductions.
- **EXPSPACE**-complete problems are *really, really* hard, even for small input lengths. But to make this precise, we should take into account the TM representation size.

Theorem 3 *There is a language L in $\mathbf{SPACE}(O(2^n))$ such that for every n, if M decides L on inputs of length n, then either M takes space $\geq 2^n$ or $|M| \geq 2^n$.*

(You can make a similar, slightly weaker, statement for REE. Either you need infeasible program size or infeasible space.)

Proof Sketch:

- The number of functions $f : \{0,1\}^n \to \{0,1\} = 2^{2^n}$
- The number of TMs of size $< 2^n$ (and space $< 2^n$) $< 2^{2^n}$.

$\Rightarrow \forall M \exists f : \{0,1\}^n \to \{0,1\}$ that is not solved by any TM of size $< 2^n$ and space $< 2^n$.

We can find such an f in $\mathbf{SPACE}(O(2^n))$. Just go over all functions in lexicographical order and see if there's a TM for it. $\square$