Computational Complexity September 20, 2018

Lecture 1 (Notes)

Lecturer: Ola Svensson *Scribes: Ola Svensson*

Disclaimer: These notes were written for the lecturer only and may contain inconsistent notation, typos, and they do not cite relevant works. They also contain extracts from the two main inspirations of this course:

1. The book *Computational Complexity: A Modern Approach* by Sanjeev Arora and Boaz Barak;
2. The course *http://theory.stanford.edu/ trevisan/cs254-14/index.html* by Luca Trevisan.

1 Introduction

- Welcome
- Ola Svensson (*ola.svensson@epfl.ch*)
- *http://theory.epfl.ch/courses/complexity*
- Goal of course: cool results, nice powerful techniques that are useful elsewhere.
- Show cool introductory slides

1.1 Grading

- $3-4$ Problem sets [60%]
- Final exam [40%]

2 Turing Machines and Deterministic Time Complexity

Let us define the Turing machines (TM) and state some standard lemmas. We refer to Chapter 1 of the textbook for their proofs.

2.1 Formal definition of a Turing Machine

A TM M is described by a tuple $(\Gamma, \mathcal{Q}, \delta)$ containing:

- A finite set Γ of the symbols that M's tapes can contain. We assume that Γ contains the designated "blank" symbol, denoted $\Box$; a designated "start" symbol, denoted $\triangleright$; and the numbers 0 and 1. We call Γ the *alphabet* of M.
- A finite set $\mathcal{Q}$ of possible states M's register can be in. We assume that $\mathcal{Q}$ contains a designated start state denoted q_{start} and a designated halting state, denoted q_{halt}.
- A function $\delta : \mathcal{Q} \times \Gamma^k \to \mathcal{Q} \times \Gamma^{k-1} \times \{L, S, R\}^k$ where $k \geq 2$, describing the rules M use in performing each step. This function is called the *transition function* of M (and k is the number of tapes).

If the machine is in state $q \in \mathcal{Q}$ and $(\sigma_1, \sigma_2, \ldots, \sigma_k)$ are the symbols currently being read in the k tapes and $\delta(q, (\sigma_1, \ldots, \sigma_k)) = (q', (\sigma'_2, \ldots, \sigma'_k), z)$ where $z \in \{L, S, R\}^k$, then at the next step the σ symbols in the last $k-1$ tapes will be replaced by the σ' symbols, the machine will be in state q', and the k heads will move Left, Right, or Stay in place, as given by z.

All tapes (except for the first tape that corresponds to the input) are initialized in their first location to the *start* symbol $\triangleright$ and in all other locations to the blank symbol $\square$. The input tape contains initially the start symbol $\triangleright$, a finite string x ("the input"), and the blank symbol $\square$ on the rest of its cells. All heads start at the left ends of the tapes and the machine is in the special starting state q_{start}. This is called the *start configuration* of M on input x.

Each step of the computation is performed by applying the function δ as described previously. The special halting state q_{halt} has the property that once that machine is in q_{halt}, the transition function δ does not allow it to further modify any tape or change states.

2.2 Basic facts

- *Robustness of definition:* The size of the alphabet, the number of tapes do not really matter
- Church-Turing Thesis: Any computable function can be computed by a Turing machine.
- Every binary string represents a TM (write down δ). A TM has infinitely many representations (pad with 1's).
- *Universality:* Turing Machines are expressive enough to simulate themselves: there is a universal Turing machine that can simulate any other Turing machine by only incurring a multiplicative factor in the time. More specifically, there exists a TM $\mathcal{U}$ such that for every $\alpha, x \in \{0,1\}^*$

  $$U(x, \alpha) = M_\alpha(x)\,.$$

  Moreover, if M_α halts on input x within T steps then $U(x, \alpha)$ halts within $C \cdot T \cdot \log T$ steps, where C is a constant independent [1] of x.

2.3 Deterministic time

Definition 1 *We say that a machine decides a language $L \subseteq \{0,1\}^*$ if it computes the function $f_L : \{0,1\}^* \to \{0,1\}$, where $f_L(x) = 1 \Leftrightarrow x \in L$.*

Definition 2 *Let $T : \mathbb{N} \to \mathbb{N}$ be some function. A language L is in* $\mathbf{DTIME}(T(n))$ *iff there is a Turing machine that runs in time at most $c \cdot T(n)$ for some constant $c > 0$ and decides L.*

Definition 3 (The class P)

$$\mathbf{P} = \bigcup_{c \geq 1} \mathbf{DTIME}(n^c)$$

3 Time Hierarchy

- Shortest path, addition, multiplication, sorting can all be done in polynomial time with a small exponent.
- Do you know any problem that is polynomial time solvable but requires time like n^{50}?
- So maybe $\mathbf{DTIME}(n^{10}) = \mathbf{DTIME}(n^{100})$? Not really…

[1] C depends on the number of states, alphabet size, and tapes of M_α.

3.1 Main Tool: Diagonalization

- Essentially only known tool for proving separations between complexity classes.
- The basic principle is the same as in Cantor's proof that the set of real numbers is not countable (1892).
- Used by Turing to prove the undecidability of the Halting problem (1936)
- To prove the Time Hierarchy Theorem by Hartmanis and Stearns (1965)

3.1.1 Cantor's diagonal argument

Let T be the set of all infinite sequences of binary digits.

Lemma 4 *If $s_1, s_2, \ldots, s_n, \ldots$ is any enumeration of the elements from T, then there is always an element s of T which corresponds to no s_n in the enumeration.*

Proof Given an enumeration of arbitrary members from T like e.g.

$$\begin{aligned}
s_1 &= (0, 0, 0, 0, 0, 0, 0, ...) \\
s_2 &= (1, 1, 1, 1, 1, 1, 1, ...) \\
s_3 &= (0, 1, 0, 1, 0, 1, 0, ...) \\
s_4 &= (1, 0, 1, 0, 1, 0, 1, ...) \\
s_5 &= (1, 1, 0, 1, 0, 1, 1, ...) \\
s_6 &= (0, 0, 1, 1, 0, 1, 1, ...) \\
s_7 &= (1, 0, 0, 0, 1, 0, 0, ...) \\
&\vdots
\end{aligned}$$

construct the sequence s by choosing the ith digit as complementary to the i:th digit of s_i. In the example, this yields:

$$\begin{aligned}
s_1 &= (\mathbf{0}, 0, 0, 0, 0, 0, 0, ...) \\
s_2 &= (1, \mathbf{1}, 1, 1, 1, 1, 1, ...) \\
s_3 &= (0, 1, \mathbf{0}, 1, 0, 1, 0, ...) \\
s_4 &= (1, 0, 1, \mathbf{0}, 1, 0, 1, ...) \\
s_5 &= (1, 1, 0, 1, \mathbf{0}, 1, 1, ...) \\
s_6 &= (0, 0, 1, 1, 0, \mathbf{1}, 1, ...) \\
s_7 &= (1, 0, 0, 0, 1, 0, \mathbf{0}, ...) \\
&\vdots \\
s &= (\mathbf{1}, \mathbf{0}, \mathbf{1}, \mathbf{1}, \mathbf{1}, \mathbf{0}, \mathbf{1}, ...)
\end{aligned}$$

By construction, s differs from each s_n since their nth digit differ. Hence, s cannot occur in the enumeration. ■

The above lemma implies that T is uncountable. If it was countable we can enumerate T but we just proved that that is not possible.

3.1.2 Halting problem is undecidable

Consider the language $L = \{\alpha \in \{0,1\}^* : M_\alpha(\alpha) \text{ does } \textbf{not} \text{ halt and accept}\}$. Note that, by a simple *reduction*, L can be decided if the Halting problem can be decided. (Recall that, given $\alpha, x \in \{0,1\}^*$, the Halting problem is to decide whether the TM M_α halts on input x.) Therefore, the following implies that the Halting problem is undecidable.

Lemma 5 *The language L is undecidable.*

Proof Suppose that there exists a TM M that decides L. Let α be a binary encoding of M. Then $M(\alpha)$ accepts if $M_\alpha(\alpha)$ does not halt and accept which is a contradiction since $M_\alpha = M$. Similarly if $M(\alpha)$ rejects then $M_\alpha(\alpha)$ halts and accepts which is again a contradiction.

	α_1	α_2	α_3	α_4
α_1	$M_{\alpha_1}(\alpha_1)$	$M_{\alpha_1}(\alpha_2)$	$M_{\alpha_1}(\alpha_3)$	$M_{\alpha_1}(\alpha_4)$
α_2	$M_{\alpha_2}(\alpha_1)$	$M_{\alpha_2}(\alpha_2)$	$M_{\alpha_2}(\alpha_3)$	$M_{\alpha_2}(\alpha_4)$
α_3	$M_{\alpha_3}(\alpha_1)$	$M_{\alpha_3}(\alpha_2)$	$M_{\alpha_3}(\alpha_3)$	$M_{\alpha_3}(\alpha_4)$
α_4	$M_{\alpha_4}(\alpha_1)$	$M_{\alpha_4}(\alpha_2)$	$M_{\alpha_4}(\alpha_3)$	$M_{\alpha_4}(\alpha_4)$
$\vdots$				
L				

■

3.2 Time Hierarchy – Simplified Version

Lemma 6 $\mathbf{DTIME}(n^{1.5}) \subsetneq \mathbf{DTIME}(n)$.

Proof

The proof is basically by diagonalization by simulating every turing machine $< n^{1.5}$ steps.

Consider the following Turing machine D:

On input x run for $|x|^{1.4}$ steps the Universal TM $\mathcal{U}$ to simulate the execution of M_x on x. If $\mathcal{U}$ outputs some bit $b \in \{0,1\}$ in this time then compute the opposite answer. Else output 0.

- By definition D halts in time $n^{1.4}$ and hence the language L decided by D is in $\mathbf{DTIME}(n^{1.5})$.
- Suppose toward contradiction that $L \in \mathbf{DTIME}(n)$. That is there exists a TM M and constant c such that, given any input $x \in \{0,1\}^*$, M halts within $c|x|$ steps and outputs $D(x)$.
- Time to simulate M by $\mathcal{U}$ is at most $c' \cdot c|x| \log c|x|$.
- Let x be a string representing M such that $c' \cdot c|x| \log c|x| < |x|^{1.4}$.
- Then D will obtain the output $b = M(x)$ within $n^{1.4}$ steps, but by definition of D we have $D(x) = 1 - b \neq M(x)$. Thus we have derived a contradiction.

■