Algorithms & Complexity: Lecture 4, Hierarchy theorems and a complexity zoo

Sam Barrett

February 10, 2021

1 Low-level conventions

1.1 Representation of Turing machines

- We will associate with every $\alpha \in \{0,1\}^*$ a Turing machine M_α s.t. for each Turing machine M, there are **infinitely many** α where $M = M_\alpha$.

  We will also fix a **bijection** between $\{00,11\}^*$ (a fragment of all binary strings) and the set of all TMs (for every word inside this language, there is a corresponding unique TM), we will write M_β for the TM M that $\beta \in \{00,11\}^*$ is mapped to. Here β is the canonical description of M or a *code of M* .

- We will extend our notion of M_β to $\alpha \in \{0,1\}^*$ (any binary string), we may write $\alpha = \beta\gamma$ with $\beta \in \{00,11\}^*$ and with $\gamma \in \{0,1\}^*$ being either empty or beginning with 01 or 10. In this case we also set $M_\alpha := M_\beta$. Here α is a **description** of $M = M_\beta$.

  Unpacking this:

  If we have a bitstring α and want to find the machine that it represents, you write α in the form $\beta\gamma$ and extract the initial β section.

A very useful property of the above framework is that given any $\alpha = \beta\gamma$ we can **computably extract** low-level information about $M = M_\alpha$ such as its states, transition table, alphabet etc. Extracting this information can be done in time and space **only dependant on** β, its canonical description. We can completely ignore γ in this case, thinking of it as *padding*.

Note: we know when to stop reading β as we treat the *gadgets* "01" or "10" as blanks/ end of input markers.

1.2 Constructable functions

We shall identify $\mathbb{N}$ and $\{0,1\}^*$ by some *fixed bijective coding* . This will make more sense later in the lecture. Refer back.

1.2.1 Time-constructibility convention

All functions $t : \mathbb{N} \to \mathbb{N}$ we consider are **time-constructible** meaning:

- $t(n) \geq n$
- There is a TM M computing $1^n \mapsto t(n)$ in time $t(n)$

1.2.2 Space-constructibility convention

All functions $s : \mathbb{N} \to \mathbb{N}$ we consider are **space-constructible** meaning:

- $s(n) \geq \log n$
- There is a TM M computing $1^n \mapsto s(n)$ in space $O(s(n))$

2 Universality

We have previously seen the following:

2.1 Normal form of Turing machines

Theorem 1 *Suppose M computes $f : \{0,1\}^* \to \{0,1\}$ in time $t(n)$ and space $s(n)$. Where M can have an arbitrary alphabet and any number of tapes.*

There exists a 3-tape TM $\tilde{M}$ with alphabet $\{\triangleright, \Box, 0, 1\}$ computing f,

- *in time $O(t(n)^2)$*
- *in space $O(s(n))$*

These constraints depend on M and not its description $\tilde{\beta}$

Moreover, we can compute the canonical description $\tilde{\beta}$ of $\tilde{M}$ from the canonical description β of M or any other description,α, of M for that matter (in the form $\alpha = \beta\gamma$).

2.2 An efficient universal machine, $\mathcal{U}$

For a TM M and a bitstring $x \in \{0,1\}^*$, we shall write $M(x) \in \{0,1\}^* \cup \{\uparrow\}$ for the output of M on x, if it exists, otherwise $\uparrow$ if it diverges or does not terminate.

Theorem 2 *There exists a TM $\mathcal{U}$ s.t. $\forall x \in \{0,1\}^*$ and $\alpha \in \{0,1\}^*$, we have $\mathcal{U}(x, \alpha) = M_\alpha(x)$*

*Moreover, we can also talk about its complexity, if M_α halts on x in t steps and uses s space, then $\mathcal{U}$ halts on (x, α) within $c_M t^2$ steps and $d_M s$ space, where c_M and d_M are constants depending only on $M = M_\alpha$, **not** its description α. (It will precisely depend on the canonical description of*

M β)

*Our formulation of $\mathcal{U}$ will have **5 tapes**: 1 input and 4 work tapes.*

We will now examine how $\mathcal{U}$ operates over an input (x, α):

1. Computing the normal form

   Let $\alpha = \beta\gamma$ be as defined in Section 1.1 and recall the definition of $\tilde{\beta}$ and $\tilde{M}$ from Theorem 1

   - $\mathcal{U}$ first computes $\tilde{\beta}$ from $\alpha = \beta\gamma$ and prints it onto tape 2, it only reads up to end of β
   - The first step concludes by printing a description of the start state of M on tape 3

   This step takes time and space complexity depending on β, ignoring γ.

   Usage of tapes and space complexity

   - From this stage onward, only the initial x section of the input (x, α) will be used on tape 1.

     Where we can visualise our input tape as:

$\triangleright$	x_0	x_1	$\cdots$	x_k	,	α_1	α_2	$\cdots$	α_l	

     Where the comma can be encoded as the first occurrence of our 01 or 10 gadget and our delimiter, if we encode our x in the same way as we do for our canonical descriptions β.
   - Tape 2 will become **read-only** and is used as a *lookup* table for simulating the transitions of $\tilde{M}$. Therefore, this tape uses space $|\tilde{\beta}|$
   - Tape 3 will always store a *current state*, using only as much space as the description of a state of $\tilde{M}$ (without loss in generality our space usage is $< |\tilde{\beta}|$)
   - Tapes 4 & 5 will be used as the two work tapes of $\tilde{M}$. Therefore, these tapes use only as much space as M does on its work tapes.

2. The simulation & time complexity

   Each step of $\tilde{M}$ is simulated as follows:

   - $\mathcal{U}$ inspects tape 3 to find the current state q and reads the symbols b_1, b_4, b_5 at the head-positions of tapes 1,4 and 5. This process takes no (0) time.
   - $\mathcal{U}$ scans the transition table of $\tilde{M}$ (by inspecting tape 2) to find the transition corresponding to (q, b_1, b_4, b_5). The time of this depends only on $\tilde{\beta}$
   - $\mathcal{U}$ overwrites tape 3 with the description of the new state. The time this takes depends only on $\tilde{\beta}$.

- $\mathcal{U}$ writes the appropriate symbols at the head-positions of tapes 4 and 5 before moving the heads of tapes 1,4 and 5 in the appropriate directions. This takes a single (1) time step.

$\mathcal{U}$ will halt whenever $\tilde{M}$ does, outputting the content of tape 5.

3 Diagonalisation

Theorem 3 *(Time hierarchy theorem) There is a language* $L \in \mathbf{DTIME}(t(n)^4)$ *s.t.* $L \notin \mathbf{DTIME}$

i.e. $\mathbf{DTIME}(t(n)) \subsetneq \mathbf{DTIME}(t(n)^4)$ *(one is **strictly contained within the other**)*

Where t *is arbitrary but time constructable as defined in Section 1.*

3.1 Time-sensitive diagonalisation

To perform diagonalisation in such a way as to concern ourselves with time complexity, we define a Turing machine D that does the following:

Definition 1 *(Turing machine* D*)*

- *on input* x *(*x *is a binary string* $x \in \{0,1\}^*$*), run* $\mathcal{U}$ *on* (x,x) *for* $t(|x|)^3$ *steps, we use* $t(|x|)^3$ *as it is somewhere between the time overhead for* $\mathcal{U}$ *(*$t(n)^2$*) and our states time hierarchy constraint of* $t(n)^4$*.*
- *if it halts in this time and rejects (where rejecting means it outputs 0) then accept (output 1)*
- *otherwise, reject (output 0).*

We can now define a language $L \subseteq \{0,1\}^*$ as the language that is decided by D. L is just the set of descriptions of Turing machines for which when $\mathcal{U}$ runs it on itself it rejects the appropriate amount of time.

By our construction of L we can observe that $L \in \mathbf{DTIME}(t(n)^4)$ as our machine D can only run for $t(|x|)^3$ steps.

We claim therefore, that L is the **explicit** language that separates $\mathbf{DTIME}(t(n^4))$ from $\mathbf{DTIME}(t(n))$, meaning that $L \notin \mathbf{DTIME}(t(n))$. We will prove this by contradiction.

3.1.1 Proof

Assume that $L \in \mathbf{DTIME}(t(n))$, and suppose M decides L taking $ct(n)$ steps on inputs of length n.

We now use $\mathcal{U}$ to simulate M and say it does this within $c_M ct(n)^2$ steps on inputs of length n. Where c_M depends only on M and not its description.

Let us fix $n_0 \in \mathbb{N}$ s.t. if $n \geq n_0$ then $t(n)^3 > c_M ct(n)^2$.

This is a key point, it means that there is a point in $\mathbb{N}$, n_0 where whenever n is greater than n_0 we can say that $t(n)^3$ (the number of steps $\mathcal{U}$ runs for) is greater than the number of steps our machine M is purported to take.

This is where *"foo is dependant only on bar not its description"* becomes important, the trick to *breaking* this inequality and deriving a contradiction is to let α be a description of M (which has infinitely many descriptions) with $|\alpha| \geq n_0$

We will now examine what happens when we run D with the input α that I described above.

- D runs $\mathcal{U}$ on (α, α) for $t(|\alpha|)^3$ steps as per our definition of D in Definition 1

- From our fixing above, along with our definition of α, we can say that $t(|\alpha|)^3 \geq c_M ct(|\alpha|)^2$, giving us in turn:

$$\mathcal{U}(\alpha, \alpha) = M_\alpha(\alpha) = M(\alpha)$$

  As M must halt on α **within** $ct(|\alpha|)$ steps, by our assumption of M.

- As per our definition of D, as $M(\alpha)$ halts, D must return $1 - M(\alpha)$ as it always returns the inverse.

  However, by doing so, as M was meant to decide the language described by D we have derived a contradiction by constructing a situation in which M and D disagree on an input α. Therefore, M could **not** have decided the language described by D.

By a similar proof, tracking space instead of time we can show that:

Theorem 4 *(Space hierarchy theorem) There is a language $L \in$ **SPACE**$(t(n)^2)$ s.t. $L \notin$ **SPACE**$(t(n))$*

We will not go on to prove this.

4 Consequences and the complexity *zoo*

4.1 Separations of complexity classes

Theorem 5 *We have the following:*
P $\subsetneq$ **EXP**
L $\subsetneq$ **PSPACE**

4.1.1 Proof

For both of the above statements, the $\subseteq$ case is *obvious*. However, for non-equality we have,

$$\mathbf{P} \subseteq \overbrace{\mathbf{DTIME}(2^n) \subsetneq \mathbf{DTIME}(2^{4n})}^{\text{Time hierarchy theorem}} \subseteq \mathbf{EXP}$$

This can be seen by the time-hierarchy theorem explored earlier.
We also have for space:

$$\mathbf{L} \subseteq \mathbf{SPACE}(n) \subsetneq \mathbf{SPACE}(n^2) \subseteq \mathbf{PSPACE}$$

which can equally be seen via the space-hierarchy theorem.
We now have a *zoo* of complexity classes:

$$\mathbf{L} \underbrace{\subseteq}_{\text{Lecture 2}} \mathbf{P} \underbrace{\subseteq}_{\text{obv}} \mathbf{NP} \subseteq \overbrace{\mathbf{PSPACE} \underbrace{\subseteq}_{\text{obv}} \mathbf{NPSPACE}}^{\leftarrow\text{Savitch's theorem}} \subseteq \mathbf{EXP}$$

This is all we know! Every other such problem remains open.