CS 810: Introduction to Complexity Theory 01/23/03

Lecture 02: Time and Space Hierarchies

Instructor: Jin-Yi Cai Scribe: Christopher Hudzik, Sarah Knoop, Louis Kruger

1 Introduction

1.1 Lecture Overview

The last lecture we encountered the method of diagnalization in two incarnations: Cantor's proof of uncountability of reals, and Turing's proof of the undecidability of the Halting Problem. In this lecture we will see this method of diagnalization once again providing the foundation of complexity theory—namely with more time and more space one can provably compute more. Given more time or space, Turing machines can decide more languages. We will make this more quantitative. Question: How much more time/space is is sufficient for this result?

1.2 Turing Machines

A Turing machine (TM) has two parts to it, the tape and the control. The tape is an sequence of cells extending infinitely to the left and to the right. Each cell contains one character from a finite alphabet. The control consists of a finite number of states. There is also a tape head which scans the tape and provides the communication between the finite states and the tape. At any time step, the tape head reads one tape cell. Then, depending on the current state and the character read, the TM may have a valid move consisting of a new state, a new character written on the current tape cell, and the tape head will move right or left one cell. At the beginning of a computation, the tape contents consist of all blanks except for the input, and the tape head points to the beginning of the input. Formally, a TM M is a six-tuple $(Q, q_0, F, \Sigma, \Gamma, \delta)$ defined by:

1. a finite set of states Q,
2. an initial state $q_0 \in Q$,
3. a subset of accepting states $F \subseteq Q$,
4. an input alphabet Σ,
5. a tape alphabet Γ where $\Sigma \subset \Gamma$ and there is a blank symbol B such that $B \in \Gamma \setminus \Sigma$, and
6. a transition function δ such that δ is a partial function and $\delta : (Q \setminus F) \times \Gamma \to Q \times \Gamma \times \{L, R\}$.

The input of the transition function is the current state and the current tape symbol. The output of the transition function is the next state, the new tape symbol to be written on the current tape cell, and the direction that the tape head will move. One application of the transition function is called one step of the TM. The transition function δ can be thought of as a list of five-tuples (q, A, q', B, D) where $q \in (Q \setminus F), A \in \Gamma, q' \in Q, B \in \Gamma$, and $D \in \{L, R\}$. One step of a TM is the result of one application of the transition function.

This definition characterizes a deterministic TM, since from any given state, δ can map to at most one next configuration. Allowing δ to not be a function but a relation will give a non-deterministic TM. In this case, δ can describe multiple "next configurations." Subsequent propositions refer to this deterministic model, though can be extend to the non-deterministic model.

Define a configuration of a TM to be the current state, the current tape cell, and the tape contents. We write a configuration as $\alpha q \beta$ where q is the current state, the first character of β is the current

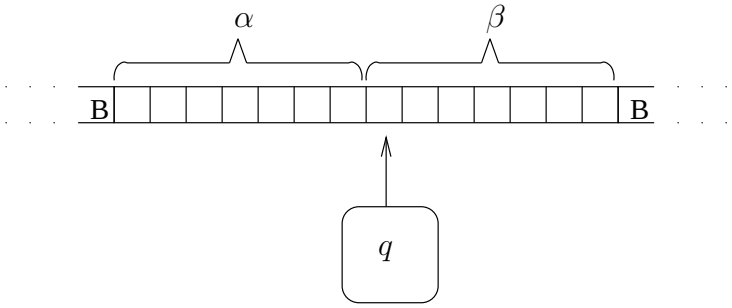

Figure 1: Configuration $\alpha q \beta$

tape cell, and $\alpha\beta$ is the contents of the tape (see Figure 1). Note that the contents of the tape are infinite, but only finitely many of the characters are not the blank symbol. Thus, let $\alpha\beta$ be some finite part of the contents of the tape such that any characters to the left or right of $\alpha\beta$ is a blank symbol, and the first character of α is not a blank and the last character of β is not a blank. (If all symbols are blank, we represent this by the empty string $\alpha\beta = \epsilon$. For convenience we can also consider an equivalence relation over Γ^*, where two sequences are equivalent iff they differ by any number of B at the beginning as well as the end of the sequences.) This information $\alpha q \beta$ is sufficient to define the next configuration. Given a TM M, let $\vdash_M$ be the binary relation defined by $\alpha q \beta \vdash_M \alpha' q \beta'$ if and only if, beginning with $\alpha q \beta$, $\alpha' q \beta'$ is the configuration of M after one step. Define $\vdash_M^*$ to be the binary relation defined by $\alpha q \beta \vdash_M \alpha' q \beta'$ if and only if, beginning with $\alpha q \beta$, $\alpha' q \beta'$ is the configuration of M after zero or more steps (reflexive and transitive closure).

The language accepted by a TM M is denoted $L(M)$ and is defined

$$L(M) \doteq \{x | (\exists \alpha, \beta \in \Gamma^*) q_0 x \vdash_M^* \alpha q_y \beta \text{ where } q_y \in F\}.$$

We say M accepts on input x iff $x \in L(M)$.

2 Time and Space Complexity Hierarchies

2.1 Constructible Functions

A function $f(n)$ is *time constructible* if $\exists$ TM $M_f, \forall n, \exists x, |x| = n$ and $M_f(x)$ halts in exactly $f(n)$ steps. A function is *fully-time constructible* if $\exists$ TM $M_f, \forall n, \forall x, |x| = n$ and $M_f(x)$ halts in exactly $f(n)$ steps. Almost all reasonable functions $\geq n$, are fully-time constructible, e.g., s n, n^k, $n^i (\log n)^j$, $2^{(\log)^k}$, 2^{n^k}, $f_i(n) = n^i + i$ are all fully-time constructible for any integer i, j, k. They are also closed under addition, multiplication, exponentiation, etc. [1] Also note that for any polynomial

[1] It is a fact in complexity theory that there exist functions which are not fully-time constructible, but we will not be concerned with that.

f, there exists a large enough i such that $n^i + i$ majorizes f.

A function $f(n)$ is *space constructible* if $\exists$ TM $M_f, \forall n, \exists x, |x| = n$ and $M_f(x)$ visits exactly $f(n)$ tape cells during its computation. A function is *fully-space constructible* if $\exists$ TM $M_f, \forall n, \forall x, |x| = n$ and $M_f(x)$ visits exactly $f(n)$ tape cells during its computation. Most nice functions $\geq \log n$ are fully-space constructible.

2.2 Bounding TM's Running Times

Given a TM M and a fully-time constructible function f, let $M_{[f]}$ denote the TM M with a clock f. This means we equip M with a concurrently running subroutine that runs exactly f steps. Thus, on input x, M is forced to halt after $f(|x|)$ steps. Thus, $M_{[f]}$ accepts on input x exactly when M accepts on input x in at most $f(|x|)$ number of steps.
We have an enumeration of TM's, $M_1, M_2, M_3 \ldots$. Define $f_i \doteq n^i + i$. Then we get a new enumeration of TM's $M_{<i,j>} \doteq M_{i[f_j]}$ for i, j positive integers. Thus we may assume an enumeration of TM's for polynomial time with an explicit clock.

2.3 DTIME

Given a TM M and input x, we define $\text{time}_M(x)$ to be the number of steps that the TM M runs on input x until halting. Then, given a positive integer n, $\text{time}_M(n)$ is defined by $\max\{\text{time}_M(x) : |x| = n\}$.

Definition 1 (DTIME). *For any time constructible function $f : \mathbb{N} \to \mathbb{N}$,*

$$DTIME[f(n)] \doteq \{L | \exists \text{ TM } M, L = L(M) \text{ and } time_M(n) \leq cf(n) \text{ for some constant } c\}$$

This definition allows for a couple assumptions:

- M halts in $f(n)$ steps. In fact, we can assume that M has an explicit clock f.
- M is a multitape TM, for the sake of robustness.

The complexity class P is defined to be $\bigcup_{i \geq 1} \text{DTIME}[n^i + i]$.

2.4 DSPACE

We define space complexity next. In order to account for sublinear space, one uses a separate read-only input tape, in addition to a read-write work tape. On the read-only input tape, the input of length n is written, but these n cells do not count toward space complexity. We only count the number of tape cells used on the read-write work tape. Thus for space complexity, the standard model is what is known as an off-line TM, which has one read-only input tape, and one read-write work tape. (On the single work tape one simulate multiple work tapes by multiple tracks–thus a larger alphabet set—without any extra space.) The space complexity of a TM is determined by the number of cells visited by the work tape head during the execution of the TM. Given a TM M and input x, we define $\text{space}_M(x)$ to be the number of tape cells that the TM M visits on input x until halting. Then, given a positive integer n, $\text{space}_M(n)$ is defined by $\max\{\text{space}_M(x) : |x| = n\}$. We know that a TM using space $S(n)$ will halt in $c^{S(N)}$ steps for some constant c or repeat a configuration because that number will bound the number of configurations given $S(n)$ tape cells, thus, never halting.

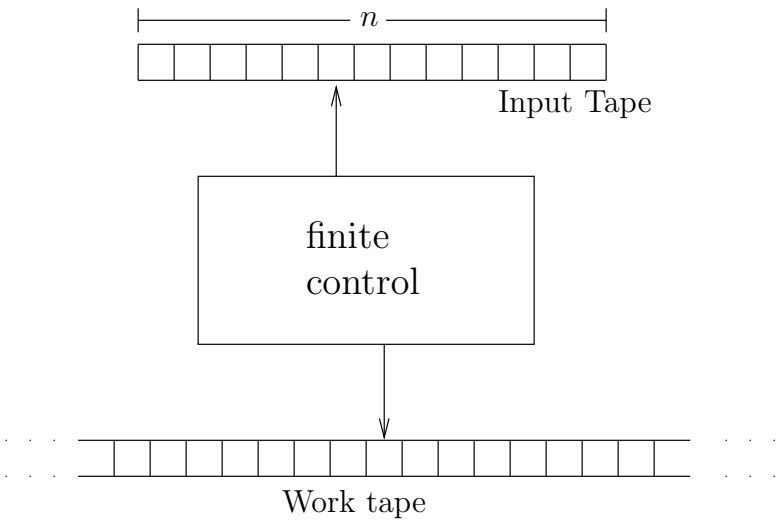

Figure 2: A TM with a separate input tape

In order to not have the input increase the space complexity, we modify our TM to have a separate read-only input tape (see Figure 2). Thus, the new TM will need two tape heads: one for the work tape, the original tape, and one for the input tape. The work tape will solely deterimine the space complexity. This allows for the possibility of sublinear space complexity. For example, there is a TM using logarithmic space to find a 3-clique of a graph. On the input tape are the vertices and edges of the graph. To find a 3-clique, the TM needs only to run through all of sets of three vertices and determine if they form a 3-clique. This requires constant space for the the vertices plus logarithmic space to keep a counter. Thus, the whole TM runs in logarithmic space. A language can be described by a TM that runs in space in $o(\log \log n)$ if and only if the language is regular.

Definition 2 (DSPACE). *For any space constructible function* $f : \mathbb{N} \to \mathbb{N}$,

$$DSPACE[f(n)] \doteq \{L | \exists \text{ TM } M, L = L(M) \text{ and } space_{_M}(n) \leq cf(n) \text{ for some constant } c\}$$

In both definitions of time and space complexity, we usually require $\text{time}_{_M}(n)$ and $\text{space}_{_M}(n) \leq cf(n)$ for all sufficiently large n.

2.5 Linear speed-up—Constant factors don't matter

Theorem 1 (Tape Compression Theorem). *Given a constant* $c > 0$*, for any space constructible function,* $S(n)$*,* $DSPACE(S(n)) = DSPACE(c \cdot S(n))$.

Proof. Let M be a TM that runs with space bound $S(n)$. The idea is to simulate M with a new TM M' that runs with space bound $c \cdot S(n)$ using the same number of tapes. Choose a positive integer m s.t. $m > 1/c$. Assume M has alphabet Γ of size k, r states $\{q_0, q_1, \ldots, q_r\}$ and t tapes. Now, we will group the tape squares of M into blocks of m squares, using an alphabet of size k^m. In addition, M' will have $r \cdot m^t$ states, where each state represents the original state q_i of the machine M, along with the position of each head within the current blocks of m squares. M' can simulate the action of M. Therefore $L(M') = L(M)$ and M' has a space bound $S(n)/m \leq c \cdot S(n)$.

□

Theorem 2 (Linear Speed-Up Theorem). *Suppose for a time constructible function, t(n), that* $\lim_{n\to\infty} t(n)/n = \infty$. *Then for any constant* $c > 0$, *DTIME*$(t(n))$ =*DTIME*$(c \cdot t(n))$.

Proof. Let M be a TM that runs in time bound $t(n)$. The idea for this proof is similar to the Tape Compression Theorem, in that we will chunk the original tapes into groups of m squares, where m is a sufficiently large number. For each group g, call its left neighboring group g_l and its right neighboring group g_r, and let $H(g, g_l, g_r)$ be the set of all sequences of moves of M starting from entering g until halting, looping, or leaving the triple of groups. The total number of such sequences are bounded by a function of m. Now we will construct a machine M'. Initially, M' encodes each group of M into a symbol onto a extra tape. Thus, M' uses 1 more tape than M. Then it simulates M by simulating each history $H(g, g_l, g_r)$ in a constant number of moves by visiting g, g_l, and g_r, then determing the sequence of moves that M would perform in H. It overwrites the symbols in g, g_l, and g_r, and continues the simulation on the next set of groups. Note that each history in $H(g, g_l, g_r)$ encodes at least m moves of M, and its simulation takes c_1 moves of M' where c_1 is a constant that is independent of m. The initial copy and encoding of the input string, followed by returning the tape head to the leftmost square takes $n + \lfloor n/m \rfloor$ moves, so M' has time bound $n + \lceil n/m \rceil + c_1 \cdot \lceil t(n)/m \rceil \leq n + n/m + c_1 \cdot t(n)/m + c_1 + 1$. If $m > c_1/c$, then for sufficiently large n, M' has time bound $\leq c \cdot t(n)$.

□

2.6 Time Hierarchy

The following proposition asserts the reduction of any multi-tape TM to a TM with only two work tapes with increasing the time complexity by a logarithmic factor. We will accept this proposition without proof.

Proposition 1. *Any multi-tape TM with running time* $t(n)$ *can be simulated by a 2-worktape TM with running time in* $O(t(n) \log t(n))$.

It is easier to see a simulation by even one tape TM with $O(t(n)^2)$. This simulation can be seen easily as follows: Divide the single tape into $2k$ tracks, and use a large alphabet set with more than 2^{2k} symbols, say. Keep on the single tape the contents of all k tapes, together with a mark for each head position. Then one step of the computation of the k-tape TM is simulated by the 1-tape TM with 2 sweeps. Note that each sweep of the tape area which has been used takes at most $t(n)$ steps. Now we will prove a time and space hierarchy theorem. Basically, given time $T(n)$ we will try to use that amount of time to kill off the next TM with a clock of $t(n)$. The extra factor of $\log t(n)$ time allows us to eventually succeed for any TM with time $t(n)$. So in order to make sure all such $t(n)$-time TMs are killed, we consider each $t(n)$-time TMs infinitely often, for ever longer inputs.

Lemma 1. *If* $T : \mathbb{N} \to \mathbb{N}$ *is a fully-time constructible function, then there exists a language L such that L is decidable in time* $O(T(n))$ *but not decidable in time* $o(T(n)/\log T(n))$. *[Sip]*

Proof. We will construct a TM M' such that $L(M')$ is decidible in time $O(T(n))$ but not decidible in time $o(T(n)/\log T(n))$ using diagonalization.

Define M' on input x by: Let $n = |x|$. Compute and store the value $T(n)/\log T(n)$ to be used as a counter. If the counter reaches 0, then reject. If x is not of the form $[M]10^*$ where $[M]$ is the encoding of some TM M. Simulate M on x, decrementing the counter for each step M needs. If M accepts, then reject. If M rejects, then accept.

Now we show that M' runs in time $O(T(n))$. In order to simulate M, M' must keep track of M's tape, states, and transition function. In order to simulate M in $O(T(n))$, M' must keep the current tape symbol, current state, and transition function information near each other. This can be accomplished by using two tapes: one to hold the tape symbols and the other to hold the transition function and the state. Proposition 1 guarantees this to be possible. During execution, M' keeps the transition function and state near the current tape symbol. Because the transition function depends only on M, copying it will only require a constant amount of time, and because the transition function, state, and current tape symbol are near each other, looking up the next move takes only constant time. Another tape is necessary to keep the counter near the current tape symbol of M. The length of the counter is $O(log(T(n))$, so each move costs $O(log(T(n))$. Therefore, the total running time is $O(T(n))$.

Last, we need to show that $L(M')$ can not be decided in time $o(T(n)/\log T(n))$. Suppose, for a contradiction, that $L(M')$ can be decided in time $t(n) \in o(T(n)/\log T(n))$ by a TM M^*. Thus, M' can simulate M^* in $c \cdot t(n))$ for some constant c not counting the counter's steps. By definition of $t(n) \in o(T(n)/\log T(n))$, $\exists n_0, \forall n > n_0, c \cdot t(n) < T(n)/\log T(n)$. Thus, for an input of length longer than n_0, the simluation of M^* will complete. Consider the input $[M^*]10^{n_0}$. The length is greater than n_0, and, hence, the simulation of M^* will complete. Therefore, by the definition of the TM M', $[M^*]10^{n_0} \in L(M') \Leftrightarrow [M^*]10^{n_0} \notin L(M^*)$, contradicting the assumption that M^* decides $L(M')$. Thus, $L(M')$ cannot be decided in $o(T(n)/\log T(n))$. □

Theorem 3 (Time Hierarchy Theorem). *If functions $t(n)$ and $T(n)$ are fully-time constructible such that*

$$\lim \inf_{n\to\infty} \frac{t(n)\log t(n)}{T(n)} = 0$$

then

$$DTIME[t(n)] \subsetneq DTIME[T(n)].$$

Proof. Let $t : \mathbb{N} \to \mathbb{N}$ be a function in $o(T(n)/\log T(n))$. Then, by Lemma 1, there exists a language L such that L is decidible in time $O(T(n))$ but not in time $t(n)$. This implies that DTIME$[t(n)] \subsetneq$ DTIME$[T(n)]$.
Also, $t(n) \in o(T(n)/\log T(n)) \Longrightarrow t(n) \in o(T(n)/\log t(n))$ because $t(n) \in o(T(n))$, and threrefore, $t(n)\log t(n) \in o(T(n))$.

□

Note: The diagonalization method used in proving the Time Hierarchy Theorem varies slightly from the previous application, for example in the halting problem. Previously, the constructed machine disagrees with the enumerated machines on a specified input, namely the index of the enumerated machine. In these proofs, we can only say that the constructed machine will eventually disagree with the enumerated machines on inputs greater than a certain length. This is not a problem because the constructed machine has infinitely many chances to disagree with each enumerated machine on inputs of increasing length. This also applies to the Space Hierarchy Theorem.

2.7 Space Hierarchy

The following proposition asserts the reduction of any multi-tape TM to a TM with only one work tape without increasing the space complexity. We will accept this proposition without proof.

Proposition 2. *Any multi-tape TM with space complexity $S(n)$ can be simulated by a 1-worktape TM in space $O(S(n))$.*

Theorem 4 (Space Hierarchy Theorem). *Suppose functions $S_1(n)$ and $S_2(n)$ are fully space constructible and $S_1(n) \geq \log n$. Then,*

$$\liminf_{n\to\infty} \frac{S_1(n)}{S_2(n)} = 0,$$

implies

$$DSPACE[S_1(n)] \subsetneq DSPACE[S_2(n)].$$

Proof. For this proof, we will construct a TM M, using the diagonalization technique, that runs in $S_2(n)$ space whose language, $L = L(M)$, is a member of DSPACE($S_2(n)$) but not of DSPACE($S_1(n)$). Enumerate all TM's, M_i, that use $S_1(n)$ space to decide their language. By Proposition 2, it is sufficient to just enumerate all one-worktape TM's M_i.
Construct a TM M that works within space $S_2(n)$ as follows on input $\langle [M_i], x\rangle$:

1. Simulate M_i on the input $\langle [M_i], x\rangle$.
2. Halt and accept if M_i rejects.
3. Halt and reject if any of the following occur:
   (a) Simulation has run for $d \cdot k^{S_2(n)}$ time and has not halted for $k = |\Gamma|$ and some constant d,
   (b) Simulation exceeds space bound of $S_2(n)$, or
   (c) M_i accepts on input $\langle [M_i], x\rangle$.

Here, $[M_i]$ denotes the encoding of machine M_i and x is string from the input alphabet sufficiently long in order to make $n = |\langle [M_i], x\rangle|$ sufficiently large. Note that the time bound condition of $d \cdot k^{S_2(n)}$ is needed in case M_i enters an infinite loop that runs in small enough space as to not exceed the $S_2(n)$ space bound. This time bound is also acceptable because it does not limit the , as explained in section 2.4. because the total number of configurations possible in space $S_2(n)$ is less than $d \cdot k^{S_2(n)}$ for some d. Therefore, if this amount of time is exceeded, then we know the machine is in an infinite loop.

$L = L(M)$ is decidable by M in space approximately $2S_2(n)$. The simulation takes $S_2(n)$ space, and the time clock uses $S_2(n)$ space. Thus L, by Theorem 1, is in DSPACE($S_2(n)$).

Now, suppose there exists a TM M_j that decides L in $S_1(n)$ space. Observe that, as a result of $\lim_{n\to\infty} S_1(n)/S_2(n) = 0$, for any $c > 0, \exists N, \forall n \geq N, c \cdot S_1(n) < S_2(n)$. Now given any c, select x, a string of input alphabet, such that $|\langle [M_j], x\rangle| > N$. Run M on the input $\langle [M_j], x\rangle$. Since the input $\langle [M_j], x\rangle$ is longer than N in length, the simulation of M_j by M will take less than $S_2(n)$ space to run to completion, as seen in previous observation. Thus, M will do the opposite of M_j on the same input. Hence, M_j does not decide $L = L(M)$, giving a contradiction to the assumption.

Therefore, DSPACE($S_1(n)$) $\subsetneq$DSPACE($S_2(n)$).

□

References

[Sip] M Sipser. *I*ntroduction to the Theory of Computation. PWS Publishing Company, 1997.