

---

# Independent Study Complexity Theory

Ryan Dougherty

---

# Table of Contents

1	Introduction & Preface .....	3
2	Review .....	4
	2.1 (Un)Decidability .....	4
	2.2 Reducibility .....	4
	2.3 Logical Theories.....	5
	2.4 Oracle TMs .....	5
	2.5 Computational Complexity .....	6
	2.6 Space Complexity .....	7
	2.7 Relativized Complexity .....	7
3	Polynomial Hierarchy, Alternating TMs.....	8
4	Boolean Circuits .....	9
5	Randomization .....	10
6	Interactive Proofs .....	11
7	Quantum Computation .....	12
8	PCP Theorem .....	13
9	Decision Trees .....	14
10	Communication Complexity .....	15
11	Algebraic Computation Models .....	16
12	Counting Complexity.....	17
13	Average-Case Complexity .....	18
14	Hardness Amplification .....	19
15	Derandomization.....	20
16	Expanders/Extractors .....	21
17	PCP and Fourier Transform .....	22
18	Parameterized Complexity .....	23

## 1 Introduction & Preface

Welcome to this series of lecture notes! The main book that the material comes from is Arora and Barak's *Computational Complexity* book [AB09]. Some material that is assumed from the reader (and is referenced in Section 2) is from Sipser's *Introduction to the Theory of Computation* book [Sip12]. We assume that the reader has a reasonable understanding of the following material:

- {Regular, Context-free, Turing-decidable, Turing-recognizable} languages, and their machine counterparts
- (Un)decidability
- Reducibility
- Recursion theorem
- Time complexity:  $\mathcal{P}$ ,  $\mathcal{NP}$ ,  $\mathcal{EXPTIME}$ , and their -complete versions
- Space complexity:  $\mathcal{PSPACE}$ ,  $\mathcal{EXPSPACE}$ ,  $\mathcal{L}$ ,  $\mathcal{NL}$ , and their -complete versions

## 2 Review

This section highlights many of the key definitions and theorems studied in a first-year graduate (or advanced undergraduate) course in complexity theory. We assume the reader knows about finite automata (DFAs/NFAs), grammars (CFGs), and Turing machines (TMs), and their respective language classes.

### 2.1 (Un)Decidability

**Definition 1.** A TM is a decider if it halts (accepts or rejects) on every input. A language  $B$  is decidable if there exists a decider  $D$  such that  $L(D) = B$ . A language  $C$  is undecidable if  $C$  is not decidable.

**Theorem 1.** The following are decidable:

- $A_{DFA} = \{\langle M, w \rangle : M \text{ is a DFA that accepts } w\}$ .
- $E_{DFA} = \{\langle M \rangle : M \text{ is a DFA whose language is empty}\}$ .
- $ALL_{DFA} = \{\langle M \rangle : M \text{ is a DFA whose language is } \Sigma^*\}$ .
- $EQ_{DFA} = \{\langle M_1, M_2 \rangle : M_1 \text{ and } M_2 \text{ are DFAs and } L(M_1) = L(M_2)\}$ .
- $A_{CFG} = \{\langle G, w \rangle : G \text{ is a CFG that generates } w\}$ .
- $E_{CFG} = \{\langle G \rangle : L(G) \text{ is empty}\}$ .

**Theorem 2.** The following are undecidable:

- $ALL_{CFG} = \{\langle G \rangle : G \text{ is a CFG and } L(G) = \Sigma^*\}$ .
- $EQ_{CFG} = \{\langle G_1, G_2 \rangle : G_1 \text{ and } G_2 \text{ are CFGs and } L(G_1) = L(G_2)\}$ .
- $A_{TM} = \{\langle M, w \rangle : M \text{ is a TM that accepts } w\}$ .

**Theorem 3.** The class of decidable languages is closed under complement.

**Definition 2.** A language  $B$  is Turing-recognizable (or recognizable) if there exists a TM that recognizes  $B$ . A language  $C$  is co-Turing-recognizable (or co-recognizable) if it is the complement of some Turing-recognizable language.

**Theorem 4.**  $A_{TM}$  is not co-recognizable.

**Theorem 5.** A language  $B$  is decidable if and only if  $B$  is recognizable and co-recognizable.

### 2.2 Reducibility

**Definition 3.** A function  $f : \Sigma^* \rightarrow \Sigma^*$  is a computable function if there exists a TM that, on input  $w$ , halts with  $f(w)$  on its tape. A language  $A$  is mapping-reducible to language  $B$ , written  $A \leq_m B$ , if there exists a computable function  $f$  such that  $w \in A$  if and only if  $f(w) \in B$ .

**Theorem 6.** If  $A \leq_m B$  and  $B$  is decidable, then  $A$  is decidable; if  $A$  is undecidable, then  $B$  is undecidable; if  $B$  is recognizable, then  $A$  is recognizable; if  $A$  is not recognizable, then  $B$  is not recognizable.

**Corollary 1.**  $HALT_{TM} = \{\langle M, w \rangle : M \text{ is a TM that halts on input } w\}$  is undecidable.

**Definition 4.** A TM's language has a property  $P$  (a subset of all TM descriptions) such that whenever  $M_1, M_2$  are TMs, and  $L(M_1) = L(M_2)$ ,  $\langle M_1 \rangle \in P$  if and only if  $\langle M_2 \rangle \in P$ . A property  $P$  is nontrivial if some TM has property  $P$  and some other TM does not.

**Theorem 7 (Rice's Theorem).** Deciding whether a TM has a nontrivial property  $P$  of its language is undecidable.

**Theorem 8.**  $EQ_{TM} = \{\langle M_1, M_2 \rangle : M_1, M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$  is undecidable; also, it is neither recognizable nor co-recognizable.

**Definition 5.** A configuration of a TM on input  $w = w_1 \cdots w_n$  in state  $q$  is  $w_1 \cdots w_{i-1}qw_i \cdots w_n$ . A computation history is a set of configurations delimited by an extra symbol #:  $\#C_1\#C_2\#\cdots\#C_\ell\#$ , where  $C_i$  logically yields  $C_{i+1}$ . An accepting computation history is one such that  $C_1$  is the start configuration, and  $C_\ell$  is an accepting one.

**Definition 6.** A linear bounded automaton (LBA) is a TM that does not allow to move the tape head past the right end of the input.

**Theorem 9.**  $A_{LBA} = \{\langle M, w \rangle : M \text{ is an LBA that accepts } w\}$  is decidable.

**Definition 7.** The Post Correspondence Problem (PCP) is a puzzle, with a given set of tiles with nonempty “top strings” and nonempty “bottom strings.” The objective is to list the tiles, repetitions allowed, such that the concatenation of the top strings of all the chosen tiles equals the same of the bottom strings.

**Theorem 10.** PCP is undecidable.

**Theorem 11 (Recursion Theorem).** Let a TM  $T$  compute a function  $t : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ . Therefore, there exists a TM  $R$  that computes a function  $r : \Sigma^* \rightarrow \Sigma^*$ , such that  $r(w) = t(\langle R \rangle, w)$  for all  $w$ . In other words, every TM can obtain their own description.

**Definition 8.** A TM  $M$  is minimal if there does not exist a TM  $N$  that has fewer states and  $L(M) = L(N)$ .

**Theorem 12.**  $MIN_{TM} = \{\langle M \rangle : M \text{ is a TM and is minimal}\}$  is not recognizable.

### 2.3 Logical Theories

**Definition 9.** A formula over some operations is atomic if  $R_i$  over variables  $x_1, \dots, x_\ell$  is a relation of arity  $\ell$ . A formula  $\phi$  is well-formed if it is atomic, a formula formed from other atomic formulas using the operations, or of the form  $\exists x[\phi_1]$  or  $\forall x[\phi_1]$  where  $\phi_1$  is a well-formed formula. A variable is bounded if it is within the scope of a quantifier, and free otherwise. A well-formed formula with no free variables is a sentence or a statement. The universe is the set of possible values for each variable, and the model specifies the universe and relations used. The theory of a model  $M$ , called  $Th(M)$ , is the set of true statements. A formula in prenex normal form is one that has all quantifiers appear first.

**Theorem 13.**  $Th(\mathbb{N}, +)$  is decidable.

**Theorem 14.**  $Th(\mathbb{N}, +, \times)$  is undecidable.

**Definition 10.** A formal proof of a statement  $\phi$  is a sequence of statements  $S_1, \dots, S_\ell$  where  $S_\ell = \phi$ , where each  $S_i$  follows logically from preceding statements and axioms (statements not requiring a proof). If all provable statements are true, then the system is sound; if all true statements are provable, then the system is complete.

**Theorem 15.**  $Provable(\mathbb{N}, +, \times) = \{\text{set of statements in } (\mathbb{N}, +, \times) \text{ that have proofs}\}$  is recognizable.

**Theorem 16.** There exists a true, but unprovable statement in  $Th(\mathbb{N}, +, \times)$ .

### 2.4 Oracle TMs

**Definition 11.** An oracle TM  $M$  is a TM with an “oracle tape” that, when the TM writes a string onto this tape, invokes the oracle (of a language  $L$ ) and decides membership of the written string in  $L$  in zero time, and returns a yes or no answer (written as  $M^L$ ). A language  $A$  is decidable relative to a language  $B$ — $A \leq_T B$ —if there is a TM  $M^B$  that decides  $A$ .  $A$  is Turing-reducible to  $B$  if and only if  $A \leq_T B$ .

**Theorem 17.**  $E_{TM} \leq_T A_{TM}$ .

**Theorem 18.**  $A'_{TM} = \{\langle M, w \rangle : M \text{ is a TM with an oracle for } A_{TM} \text{ and } M \text{ accepts } w\}$  is undecidable relative to  $A_{TM}$ .

**Definition 12.** The minimal description of a string  $x$  ( $d(x)$ ) is the shortest string  $\langle M, w \rangle$  where TM  $M$ , on input  $w$ , halts with  $x$  on the tape. The descriptive complexity of  $x$  ( $K(x)$ ) is  $|d(x)|$ .

**Theorem 19.**  $K(x) \leq |x| + c$  for a constant  $c$ .

**Theorem 20.**  $K(xx) \leq |x| + d$  for a constant  $d$ .

**Theorem 21.**  $K(xy) \leq 2 \log_2(K(x)) + K(x) + K(y) + e$  for a constant  $e$ .

**Definition 13.** A string  $x$  is incompressible if  $K(x) \geq |x|$ .

**Theorem 22.** At least half of all strings of length  $\leq n$  are incompressible.

**Theorem 23.**  $K(x)$  is not computable.

**Theorem 24.** No infinite subset of the set of incompressible strings is recognizable.

## 2.5 Computational Complexity

**Definition 14.**  $TIME(f(n))$  ( $NTIME(f(n))$ ) is the set of languages decidable within  $O(f(n))$  steps on a single-tape deterministic (nondeterministic) TM.  $\mathcal{P} = \bigcup_{k \geq 0} TIME(n^k)$ ,  $\mathcal{NP} = \bigcup_{k \geq 0} NTIME(n^k)$ . Decision on a NTM has that every computation branch halts, time is the number of transitions on the longest computation path, and space is the maximum number of cells visited on any computation path.

**Theorem 25.** The following are members of  $\mathcal{P}$ :

- All regular languages
- All context-free languages
- $PATH = \{\langle G, s, t \rangle : G \text{ is an undirected graph having a path from } s \text{ to } t\}$

**Definition 15.** A verifier is a TM that accepts a string  $w$  and a certificate  $c$ , and verifies whether  $c$  is valid.

**Theorem 26.**  $\mathcal{NP}$  can also be defined as the set of languages with a polynomial-time verifier.

**Definition 16.** A language  $A$  is polynomial-time reducible to a language  $B$ — $A \leq_p B$ —if the reduction takes polynomial time.

**Theorem 27.** Suppose  $A \leq_p B$ . If  $B \in \mathcal{P}$ , then  $A \in \mathcal{P}$ ; if  $A \notin \mathcal{P}$ , then  $B \notin \mathcal{P}$ .

**Definition 17.** A boolean formula  $\phi$  in conjunctive normal form is one that is a conjunction of clauses, and each clause is a disjunction of literals. A formula in 3CNF has  $\leq 3$  literals per clause. A formula is satisfiable if there exists an assignment to the variables to make the formula true.

**Theorem 28.**  $3SAT = \{\langle \phi \rangle : \phi \text{ is a 3CNF formula that is satisfiable}\} \in \mathcal{NP}$ , and  $3SAT \leq_p CLIQUE = \{\langle G, k \rangle : G \text{ is a graph with a } k\text{-clique}\}$ .

**Definition 18.** A language  $B$  is  $\mathcal{NP}$ -complete if  $B \in \mathcal{NP}$ , and for every  $A \in \mathcal{NP}$ ,  $A \leq_p B$ . If only the second condition is true, then  $B$  is  $\mathcal{NP}$ -hard.

**Theorem 29.** The following are  $\mathcal{NP}$ -complete:

- $3SAT$  (the Cook-Levin theorem)
- $CLIQUE$
- $INDSET$  (same as  $CLIQUE$  but no edges between vertices)
- $VERTEX COVER$  (whether there exists a subset of vertices of size  $\leq k$  such that every edge involves a vertex in the subset)
- $HAMPATH$  (whether a directed graph contains a directed path through every vertex exactly once)
- $UHAMPATH$  (undirected version of  $HAMPATH$ )

## 2.6 Space Complexity

**Definition 19.**  $\mathcal{PSPACE} = \bigcup_{k \geq 0} \text{SPACE}(n^k)$ ,  $\mathcal{NPSPACE} = \bigcup_{k \geq 0} \text{NSPACE}(n^k)$ .

**Theorem 30.** For  $f(n) \geq n$ ,  $\text{TIME}(f(n)) \subseteq \text{SPACE}(f(n))$ .

**Theorem 31 (Savitch's Theorem).** For  $f(n) \geq n$ ,  $\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n))$ .

**Corollary 2.**  $\mathcal{PSPACE} = \mathcal{NPSPACE}$ .

**Definition 20.** A language  $B$  is  $\mathcal{PSPACE}$ -complete if  $B \in \mathcal{PSPACE}$ , and for every  $A \in \mathcal{PSPACE}$ ,  $A \leq_p B$ . If only the second condition is true, then  $B$  is  $\mathcal{PSPACE}$ -hard.

**Theorem 32.** The following are  $\mathcal{PSPACE}$ -complete:

- $\text{TQBF} = \{\langle \psi \rangle : \psi \text{ is a true quantified boolean formula}\}$  (i.e., of the form  $\psi = Q_1 x_1 \cdots Q_n x_n \phi(x_1, \dots, x_n)$  where the  $Q_i \in \{\exists, \forall\}$ ).
- $\text{FORMULA-GAME}$  (a 2-player version of  $\text{TQBF}$ , where players take turns choosing values for the variables in order)
- $\text{Generalized Geography}$  (a directed graph where each vertex is a string, and each edge has the next string start with the same letter as the previous one)
- $A_{\text{LBA}}$

**Definition 21.**  $\mathcal{L} = \text{SPACE}(\log(n))$ ,  $\mathcal{NL} = \text{NSPACE}(\log(n))$ .

**Definition 22.** A log-space transducer is a deterministic TM with read-only input, write-only output, and a read-write work tape, and the space it uses is equal to the length of the non-blank portion of the work tape +  $\log(\text{size of input}) + \log(\text{size of output})$ . A language  $A$  is log-space reducible to a language  $B$  if there is a log-space transducer that computes a function  $f$  for which  $w \in B$  if and only if  $f(w) \in A$ . A language  $B$  is  $\mathcal{NL}$ -complete if  $B \in \mathcal{NL}$ , and for every  $A \in \mathcal{NL}$ ,  $A \leq_L B$ . If only the second condition is true, then  $B$  is  $\mathcal{NL}$ -hard.

**Theorem 33.**  $\text{PATH} = \{\langle G, s, t \rangle : G \text{ is a directed graph with a directed } s - t \text{ path}\}$  is  $\mathcal{NL}$ -complete, and  $\text{coNL}$ -complete.

**Corollary 3.**  $\mathcal{NL} = \text{coNL}$

**Definition 23.** A function  $f$  is space-constructible if there is a TM that computes the function mapping  $1^n$  (in unary) to  $f(n)$  (in binary) in  $O(f(n))$  space.

**Theorem 34 (Space Hierarchy Theorem).** If  $f$  is a space-constructible function, then there exists a language that can be decided in  $O(f(n))$  space, but not in  $o(f(n))$  space.

**Corollary 4.**  $\mathcal{PSPACE} \neq \mathcal{EXPSPACE}$

**Corollary 5.**  $\mathcal{NL} \neq \mathcal{PSPACE}$

**Definition 24.** A function  $f$ , which is  $\Omega(n \log(n))$ , is time-constructible if there is a TM that computes the function mapping  $1^n$  (in unary) to  $f(n)$  (in binary) in  $O(\frac{f(n)}{\log(n)})$  time.

**Theorem 35 (Time Hierarchy Theorem).** If  $f$  is a time-constructible function, then there exists a language that can be decided in  $O(f(n))$  time, but not in  $o(\frac{f(n)}{\log(f(n))})$  time.

**Corollary 6.**  $\mathcal{P} \neq \mathcal{EXPTIME}$

**Corollary 7.** For any  $1 < c < d$ ,  $\text{TIME}(n^c) \neq \text{TIME}(n^d)$ .

**Definition 25.** A language  $B$  is  $\mathcal{EXPSPACE}$ -complete if  $B \in \mathcal{EXPSPACE}$ , and for every  $A \in \mathcal{EXPSPACE}$ ,  $A \leq_p B$ . If only the second condition is true, then  $B$  is  $\mathcal{EXPSPACE}$ -hard.

**Theorem 36.**  $\text{EQ}_{\text{REG}\uparrow} = \{\text{regular expressions with exponentiation}\}$  is  $\mathcal{EXPSPACE}$ -complete.

## 2.7 Relativized Complexity

**Definition 26.**  $\mathcal{P}^A$  (resp.,  $\mathcal{NP}^A$ ) =  $\{L : L \text{ is decided by an oracle TM with an oracle for } A \text{ in deterministic (nondeterministic) polynomial time}\}$ .

**Theorem 37.** There exist oracles  $A, B$  such that  $\mathcal{P}^A = \mathcal{NP}^A$ , and  $\mathcal{P}^B \neq \mathcal{NP}^B$ .

### 3 Polynomial Hierarchy, Alternating TMs



## 4 Boolean Circuits

## 5 Randomization

## 6 Interactive Proofs

## 7 Quantum Computation

## 8 PCP Theorem

## 9 Decision Trees

## 10 Communication Complexity

## 11 Algebraic Computation Models



## 12 Counting Complexity

## 13 Average-Case Complexity

## 14 Hardness Amplification

## 15 Derandomization

## 16 Expanders/Extractors

## 17 PCP and Fourier Transform

## 18 Parameterized Complexity

## References

- [AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.
- [Sip12] Michael Sipser. *Introduction to the Theory of Computation*. Course Technology, 2012.