Definition 1 (DFA (Def. 1.5)) *Deterministic finite automaton is defined by 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where*

- *Q is a finite set of states*
- *Σ is a finite alphabet*
- *$\delta : Q \times \Sigma \to Q$ is the transition function*
- *q_0 is the start state*
- *$F \subseteq Q$ is the set of accept states*

Definition 2 (NFA (Def. 1.37)) *A nondeterministic finite automaton (NFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where*

- *Q is a finite set of states*
- *Σ is a finite alphabet*
- *$\delta : Q \times \Sigma_\epsilon \to \mathcal{P}(Q)$ is the transition function*
- *$q_0 \in Q$ is the start state*
- *$F \subseteq Q$ is the set of accept states.*

Definition 3 (RE (Def. 1.52)) *R is a* regular expression *(RE) if*

1. *$R = a$ for some a in the alphabet Σ,*
2. *$R = \epsilon$,*
3. *$R = \emptyset$*
4. *$R = (R_1 \cup R_2)$, where R_1 and R_2 are regular expressions*
5. *$R = (R_1 \circ R_2)$, where R_1 and R_2 are regular expressions, OR*
6. *$R = (R_1^*)$, where R_1 is a regular expression.*

Note:

- ϵ represents a language with one string; **empty string**
- $\emptyset$ represents a language with **no strings**
- Regular expressions get defined recursively!

Theorem 1 (Pumping Lemma (Def. 1.70)) *If A is a regular language, then there is a number p, referred to as the* pumping length*, where if s is* any *string in A with $|s| \geq p$, then s may be divided into three parts, $s = xyz$, satisfying the following conditions:*

1. *For each $i \geq 0$, $xy^iz \in A$*
2. *$|y| > 0$*
3. *$|xy| \leq p$*

Definition 4 (CFG (Def. 2.2)) *A* **context-free grammar** *is a 4-tuple (V, Σ, R, S), where*

- *V is a finite set called the variables,*
- *Σ is a finite set, disjoint from V, called the terminals,*
- *R is a finite set of rules, with each rule being a variable and a string of variables and terminals, and*
- *$S \in V$ is the start variable*

Definition 5 (Def. 2.8) *A context-free grammar is in* **Chomsky normal form** *if every rule is of the form*

$$A \to BC$$
$$A \to a$$

where a is any terminal and A, B, and C are any variables – except that B and C may not be the start variable. In addition, we permit the rule $S \to \epsilon$, where S is the start variable.

Definition 6 (Def. 2.13) *A pushdown automata is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$, where*

- *Q is the finite set of states*
- *Σ is the finite input alphabet*
- *Γ is the finite stack alphabet*
- *$\delta : Q \times \Sigma_\epsilon \times \Gamma_\epsilon \to \mathcal{P}(Q \times \Gamma_\epsilon)$ is the transition function*
- *$q_0 \in Q$ is the start state*
- *$F \subseteq Q$ is the set of accept states*

Theorem 2 (Thm. 2.34: Pumping lemma for CFL) *If A is a context-free language, then there is a number p (the pumping length) where, if s is any string in A of length at least p, then s may be divided into five pieces, $s = uvxyz$, satisfying the conditions*

1. *$uv^ixy^iz \in A$ for each $i \geq 0$*
2. *$|vy| > 0$*
3. *$|vxy| \leq p$.*

Definition 7 (Turing machine TM (Def. 3.3)) *Turing machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$:*

- *Q is the finite set of states*
- *Σ is the finite input alphabet, not containing the blank symbol $\sqcup$*
- *Γ is the finite tape alphabet, $\sqcup \in \Gamma$ and $\Sigma \subset \Gamma$*
- *$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function*
- *$q_0, q_{accept}, q_{reject} \in Q$ are the start, accept and reject states, $q_{accept} \neq q_{reject}$*

Definition 8 (Def. 3.5) *Language L is* **Turing-recognizable** *if Turing machine M exists that recognizes L.*

Definition 9 (Def. 3.6) *A language is* **Turing-decidable** *(or decidable) if a Turing Machine exists that decides it.*

Definition 10 *Set $\mathbb{X}$ is* countable *if it is finite, or a bijection between $\mathbb{X}$ and $\mathbb{N} = \{1, 2, \ldots\}$ exists*

Theorem 3 (Theorem 4.22) *A language is decidable iff it is Turing-recognizable and co-Turing-recognizable.*

Definition 11 *A function $f : \Sigma^* \to \Sigma^*$ is a computable function if some TM, on every input w, halts with just $f(w)$ on its tape.*

Definition 12 (Def. 5.20) *Language A is mapping reducible to language B, written $A \leq_m B$, if there is a computable function $f : \Sigma^* \to \Sigma^*$, where for every w, $w \in A \iff f(w) \in B$. The function f is called the reduction from A to B.*

Definition 13 (Def. 7.7) *Let $t : \mathbb{N} \to \mathbb{R}^+$ be a function. Define the* **time complexity class**, *TIME$(t(n))$, to be the collection of all languages that are decidable by an $O(t(n))$ time Turing machine.*

Definition 14 (Def. 7.12) *P is the class of languages that are decidable in polynomial time on a deterministic single-tape Turing machine,*

$$P = \bigcup_k \mathit{TIME}(n^k)$$

Definition 15 (Hamiltonian Path problem) *HAMPATH* $\triangleq$ $\{\langle G, s, d\rangle \mid$ *G is a directed graph with h. p.* $s \to d\}$

Definition 16 (Def. 7.19) *NP is the class of languages that have polynomial time verifiers.*

Theorem 4 (Thm. 7.20) *A language is in NP iff it is decided by some nondeterministic polynomial time Turing machine.*

Definition 17 (Subset-Sum Problem) *Given n items with weights w_i, is there a subset of items with a total weight of W.*

Definition 18 (Def. 7.28) *A function $f : \Sigma^* \to \Sigma^*$ is a* **polynomial time computable function** *if some polynomial time Turing machine M exists that halts with just $f(w)$ on its tape, when started on any input w.*

Definition 19 (Def. 7.29) *Language A is* **polynomial time (mapping) reducible** *to language B, written $A \leq_P B$, if a polynomial time computable function $f : \Sigma^* \to \Sigma^*$ exists, where for every w,*

$$w \in A \iff f(w) \in B.$$

The function f is called the polynomial time reduction of A to B.

Definition 20

$$SAT = \{\langle \phi \rangle \mid \phi \text{ is a satisfiable Boolean formula}\}$$

Definition 21 (Def. 7.34) *A language B is NP-complete if it satisfies two conditions:*

1. *B is in NP*
2. *Every A in NP is polynomial time reducible to B*

Definition 22 (NP-hard) *A language B is NP-hard if every $A \in$NP is polynomial time reducible to B.*

Corollary 5 *A language B is NP-complete if it is (i) NP and (ii) NP-hard.*

Decision problem	Function problem	Problem name	Problem description	P	NP-complete	NP-hard
	✓	Shortest path	Find a shortest path from i to j	✓		
	✓	Min. spanning tree	Find the minimum spanning tree	✓		
✓		Vertex cover	Does G have a size k vertex cover		✓	
	✓	Minimum vertex cover	Determine a smallest possible v.c.			✓
✓		Clique	Does G contain a clique of size k		✓	
	✓	k-clique	Determine a size k clique in G		✓	
	✓	Maximum clique	Determine a largest clique in G			✓
✓		Independent set decision	Does G have a size k i.s.		✓	
	✓	Maximum independent set	Determine a largest possible i.s.			✓

- *Decision problem*: output is `True` or `False`
- *Function problem*: output is a `string`, e.g., a list of nodes
  - Instead of P and NP, we have FP and FNP, ...

Table 1: Time complexity of common graph problems.

Definition 23 (Def. 8.1) *Let M be a deterministic TM that halts on all inputs. The* **space complexity** *of M is the function $f : \mathbb{N} \to \mathbb{N}$, where $f(n)$ is the maximum number of tape cells that M scans on any input of length n.*

If M is a NTM wherein all branches halt on all inputs, we define the space complexity to be the maximum number of tape cells M scans on any branch for any input of length n.

Theorem 6 (Savitch's theorem (Thm. 8.5)) *For any function $f : \mathbb{N} \to \mathbb{R}^+$, where $f(n) \geq n$*

$$NSPACE(f(n)) \subseteq SPACE(f^2(n)).$$

Definition 24 (PSPACE and NPSPACE) *PSPACE is the class of languages that are decidable in polynomial space*

$$PSPACE = \cup_k SPACE(n^k)$$

and similarly for the NPSPACE.

Definition 25 (Probabilistic Turing Machine (PTM)) *Probabilistic TMs are like NTM, but*

1. *Single branch with two possible moves in every step*
   - *50:50 probability, known as the coin-flip*
   - *independent events!*
2. *Outcome quantified using probabilities*

$$\mathbf{P}\{\texttt{accept}\} = \sum_{b \in \mathcal{B}_A(w)} 2^{-k_b},$$

$$\mathbf{P}\{\texttt{reject}\} = \sum_{b \in \mathcal{B}_R(w)} 2^{-k_b},$$

where $\mathcal{B}_A(w)$ and $\mathcal{B}_R(w)$ denote the sets of branches with accepting and rejecting final state, and k_b is the number of steps in branch b.

Definition 26 *Time complexity of PTM is according to the worst case of all branches.*

Definition 27 (Probabilistic TM and Decidability) *Probabilistic TM M decides on language A* with error probability *ϵ iff*

1. *$\mathbf{P}\{\texttt{accept}\} \geq 1 - \epsilon$ for all $w \in A$*
2. *$\mathbf{P}\{\texttt{reject}\} \geq 1 - \epsilon$ for all $w \notin A$*

Definition 28 (Class BPP) *Language $A \in BPP$ if there is a probabilistic polynomial time TM M (PPT-TM) that decides on language A with error probability $\epsilon = 1/3$.*

		DFA	CFG	TM
Accept	A_x	✓	✓	✗
Empty	E_x	✓	✓	✗
Equivalent	EQ_x	✓	✗	✗
Halting	$HALT_{TM}$			✗
Regular L.	$REGULAR_{TM}$			✗

Table 2: Decidability results for DFAs, PDAs, and TMs

Problem	context
SAT and 3SAT	boolean expressions
3COLOR	graph node coloring
CLIQUE	size k cliques in a graph
HAMPATH	Hamiltonian path from s to d
SUBSET-SUM	list of integers and their subsets

Table 3: Some NP-complete problems.