REVIEW: COMPUTABILITY THEORY

The language of an automaton is the set of all input strings that the automaton accepts:
the language is recognized by the automaton.

Deterministic Finite Automata (DFAs)

A DFA is made up of five different parts: **Q, Σ, δ, q_0,** and **F**.

1) **Q** is the finite set of states (the single-circles and double-circles).
2) **Σ** is the finite set of possible inputs (called the alphabet).
3) **δ** is the collection of rules for the transitions from a state to another state – possibly itself – given any input from the input alphabet. (These are shown by the transition arrows, or the transition table, or as a function.)
4) **q_0** is the single start state that the DFA begins in (shown by the arrow from nowhere).
5) **F** is the set of the accept states (those shown by the double-circles). There may be one, many, or no accept states. Note: The reject states are all the other single-circle states, and of course you can't have more accept states than total states.

A DFA accepts an input if it's in an accept state after processing that input, and rejects otherwise.

Nondeterministic Finite Automata (NFAs)

An NFA is made up of five different parts: **Q, Σ, δ, q_0,** and **F** defined as for a DFA except:
δ is the collection of rules for the transitions from a state and alphabet symbol *to set of states* (not a single state as in DFAs).

An NFA transition to multiple states nondeterministically splits into a branch for each state.
An NFA transition to no states dies.
An NFA accepts an input if *at least one of its branches* is in an accept state after processing the input, and rejects otherwise.

Regular Operations:

Union	$A \cup B = \{x \mid x \in A \text{ or } x \in B\}$
Concatenation	$A \circ B = AB = \{xy \mid x \in A \text{ and } x \in B\}$
Star	$A^* = \{x_1 x_2 x_3 \ldots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$

Regular Expressions

R is a regular expression if R is:

1) $a \in \Sigma$
2) ε
3) φ
4) $(R_1 \cup R_2)$
5) $(R_1 \circ R_2)$
6) R_1^*

(where R_1 and R_2 are already regular expressions)

Notes: $\varepsilon \in A^*$ no matter what A is
$R \cup \varphi = R$
$R \circ \varepsilon = R$

NFAs, DFAs GNFAs, and regular expressions are equivalent in power (you can always convert between them and recognize the same language). They describe regular languages, which are closed under union, concatenation, and star.

Pumping Lemma for Regular Languages
If A is a regular language, then there is a number p (the pumping length) where if s is any string in the language of length at least p, then s may be divided into three sections s=xyz satisfying:

1) $|y| > 0$ (y is not the empty string)
2) $|xy| <= p$ (the first 2 sections have length not more than p)
3) $xy^kz \in A$ for every k=0, 1, 2, 3,… (can pump y up or down)

To show a language L is non-regular, assume for contradiction that L is regular. Then show an $s \in L$ of length at least p that you can't divide into s=xyz satisfying the 3 requirements no matter how you split it. This is a contradiction, so L must be non-regular after all.

Context Free Grammars (CFGs)
A CFG is made up of 4 parts:

1) A finite set of variables
2) A start variable
3) A finite set of terminals (disjoint from the variables set)
4) A finite set of rules taking a variable to other variables and/or terminals

x yields y if it follows only one rule to do the transformation
x derives y if it follows one or more steps (rules) to do the transformation
The language of a CFG is all strings derived from the start variable.
An ambiguous grammar has multiple different parse trees to generate a string.

Pushdown Automata (PDAs) = NFAs with a single stack

PDAs and CFGs are equivalent in power. They describe context-free languages (CFLs), which are closed under union, concatenation, and star (but not complementation or intersection)

Pumping Lemma for Context-Free Languages
If A is a CFL, then there is a number p (the pumping length) where if s is any string in the language of length at least p, then s may be divided into three sections s=uvxyz satisfying:

1) $|vy| > 0$ (not both v and y are the empty string)
2) $|vxy| <= p$ (the length of the middle 3 parts is not more than p)
3) $uv^kxy^kz \in A$ for every k=0, 1, 2, 3,… (can pump v and y up and down)

To show a language A is non-context-free, assume for contradiction that A is a CFL. Then show an $s \in A$ of length at least p that you can't divide into s=uvxyz satisfying the 3 requirements no matter how you split it. This is a contradiction, so A must be non-context-free after all.

A **Turing-recognized language** is *recognized* by a Turing machine called a recognizer.
On input strings in the language, this TM halts and accepts.
On input strings *not* in the language, this TM rejects by *either* halting and moving into a reject state, *or* by looping forever.
Closed under union, concatenation, star, and intersection (but not complementation).

A **decidable language** is a *decided* by a Turing machine called a decider.

On input strings in the language, this TM halts and accepts.

On input strings *not* in the language, the TM rejects by halting and moving into a reject state.

It never loops forever.

Every decidable language is also Turing-recognizable (but not vice versa).

Closed under union, concatenation, star, intersection, and complementation.

An **undecidable language** is not decided by any Turing machine.

Common proof techniques

Prove a language is **Turing-recognizable** by giving a valid Turing machine that recognizes it.

Prove a language is **decidable** by giving a valid Turing machine that decides it.

Prove a language is **undecidable** by assuming for contradiction that the language is decidable by some TM R, then using R to construct a new TM S that ends up deciding A_{TM} (which is impossible, since A_{TM} is undecidable).