1 Pre-TM Definitions

- **DFA:** A 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where Q is a finite set of states, Σ is a finite alphabet, $\delta : Q \times \Sigma \to Q$ is the transition function, $q_0 \in Q$ is the start state, $F \subseteq Q$ are the accept states.

  A DFA M accepts $w = w_1 \cdots w_n \in \Sigma^*$ if there are states $r_0, r_1, \ldots, r_n \in Q$ with $r_0 = q_0$, $r_n \in F$ and $r_{i+1} = \delta(r_i, w_{i+1})$.
- **Regular:** A language L is regular if there is a DFA D with $\mathcal{L}(D) = L$.
- **NFA:** A 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where Q is a finite set of states, Σ is a finite alphabet, $\delta : Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)$ is the transition function, $q_0 \in Q$ is the start state, $F \subseteq Q$ are the accept states.

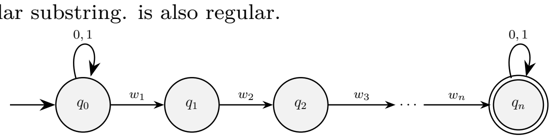
  A NFA M accepts $w = y_1 \cdots y_n \in \Sigma_\varepsilon^*$ if there are states $r_0, r_1, \ldots, r_n \in Q$ with $r_0 = q_0$, $r_n \in F$ and $r_{i+1} \in \delta(r_i, w_{i+1})$.
- For a language L, there is a DFA D with $\mathcal{L}(D) = L$ if and only if there is an NFA N with $\mathcal{L}(N) = L$.
- **Regular Closure:** If A, B are regular, then $A \cup B$, $A \cap B$, $A \circ B$, A^*, and $\overline{A}$ are all regular.
- **(Regular) Pumping Lemma:** If L is a regular language, then there is a number p such that for all $s \in L$ with $|s| \geq p$, we may write $s = xyz$ with
  1. $xy^iz \in L$ for all $i \geq 0$
  2. $|y| > 0$, and
  3. $|xy| \leq p$.
- **CFG** A 4-tuple (V, Σ, R, S) where V is a finite set of variables, Σ is disjoint from V and is finite set of terminals, R is set of rules of the form $v \to \sigma$ for $v \in V$ and $\sigma \in (V \cup \Sigma)^*$, and $S \in V$ is the start variable.

  We say $\alpha A \beta \Rightarrow_G \alpha \gamma \beta$ is a derivation in G if $A \to \gamma$ is a rule in R. We say $A \Rightarrow_G^* \gamma$ if A derives γ in zero or more steps. We say G accepts w if $S \Rightarrow_G^* w$.
- **Context-free:** A language L is context-free if there is a CFG G such that $\mathcal{L}(G) = L$.
- Every regular language is context-free.
- **Ambiguity:** A string is generated ambiguously if there are two or more derivations of the string. A regular expression/CFG is ambiguous if it generates strings ambiguously.
- **(Context-free) Pumping Lemma:** If L is a context-free language, then there is a number p such that for all $s \in L$ with $|s| \geq p$, we may write $s = uvxyz$ with
  1. $uv^ixy^iz \in L$ for all $i \geq 0$
  2. $|vy| > 0$, and
  3. $|vxy| \leq p$.
- **Context-free Closure:** If A, B are context-free, then $A \cup B$, $A \circ B$, and A^* are context-free. If A is context-free and B is regular, $A \cap B$ is context-free.
- **PDA:** A 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where Q is a finite set of states, Σ is a finite alphabet, Γ is a finite set of stack symbols, $\delta : Q \times \Sigma \times \Gamma \to \mathcal{P}(Q \times \Gamma)$ is the transition function, $q_0 \in Q$ is the start state, $F \subseteq Q$ are the accept states.

  A PDA functions like an NFA but with a stack. $\delta(q, a, \alpha) = (q', \beta)$ means in state q we read a and pop α from the top of the stack and go to state q' and push β to the top of the stack. Note if $\alpha = \varepsilon$ we don't read or pop from the stack, if $\beta = \varepsilon$ we don't push to the stack. We accept as in an NFA.
- For a language L, there is a CFG G with $\mathcal{L}(G) = L$ if and only if there is a PDA P with $\mathcal{L}(P) = L$.
- **Algorithmic Aspects:** For a DFA M we can check if $\mathcal{L}(M) = \emptyset$. For a CFG G we can check if $\mathcal{L}(G) = \emptyset$. For two DFA M_1, M_2 we can check if $\mathcal{L}(M_1) = \mathcal{L}(M_2)$. For two CFGs G_1, G_2 checking if $\mathcal{L}(G_1) = \mathcal{L}(G_2)$ is undecidable, but checking if $\mathcal{L}(G_1) \neq \mathcal{L}(G_2)$ is Turing recognizable (possibly infinite time).

2 Regular Examples

- NFA for $L = \{x \in \{0,1\}^* : w \text{ is a substring of } x\}$ for some $w \in \{0,1\}^*$. Note that by complementation the language of strings not containing a particular substring. is also regular.



- $L = \{0^k1^k : k \in \mathbb{N}\}$ is not regular.

  Proof. Let p be the pumping length and $s = 0^p1^p = xyz$. Then by (3) xy is a substring of 0^p, so $xy^2z \in L$ has $p + |y| > p$ 0's (with $|y| \geq 0$ by (2)) and p 1's. ☐
- $L = \{x \in \{0,1\}^* : x \text{ has the same number of 0's and 1's}\}$ is not regular. Proof is similar to above.
- $L = \{0^i1^j : i > j\}$ is not regular. Proof is similar to above.
- $L = \{0^k : k \text{ is prime}\}$ is not regular.

  Proof. Let p be the pumping length and $s = 0^t = xyz$ for some prime $t \geq p$. Let $r := |y| > 0$. Then $xy^{t-r}z \in L$ has length $|xz| + |y^{t-r}| = (t - r) - |y|(t - r)$ which is not prime. ☐
- $L = \{x \in \{0,1\}^* : \exists k \geq 0, x = 1^{2^k}\}$ is not regular.

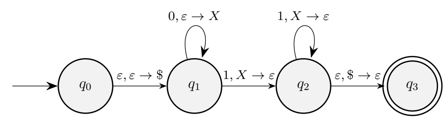
  Proof. Let p be the pumping length, $k = \lfloor \log_2 p \rfloor + 1$, and $s = 1^{2^k} = xyz$. Then $xy^2z \in L$ but $|xy^2z| > |xyz| = 2^k$ and $|xy^2z| \leq |xyz| + |xy| \leq 2^k + p < 2^{k+1}$ since $2^k > p$. ☐
- $L = \{w \in \{0,1\}^* : w = w^R\}$, language of palindromes is not regular.

  Proof. Let p be the pumping length and $s = 0^p10^p = xyz$. Then xy is a substring of 0^p by (3). So $xy^2z = 0^{p+|y|}b0^p \in L$ is clearly not a palindrome since $p + |y| > p$ by (2). ☐

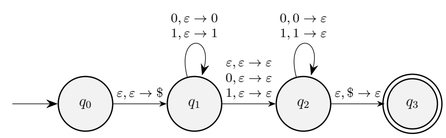
3 Context-free Examples

- CFG for $L = \{x \in \{0,1\}^* : x \text{ has the same number of 0's and 1's}\}$.

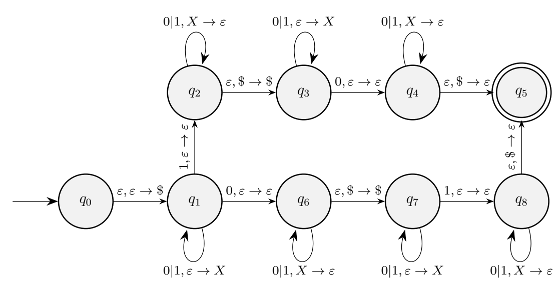
  Proof. $G = (\{S\}, \{0,1\}, R, S)$ with R being $S \to 0S1|1S0|SS|\varepsilon$. Prove $\mathcal{L}(G) \subseteq L$ by induction on ℓ being the length of the shortest deriving path of x. Prove $L \subseteq \mathcal{L}(G)$ by induction on $|x|$. ☐
- CFG for $L = \{0^k1^k : k \in \mathbb{N}\}$. Consider $G = (\{S\}, \{0,1\}, R, S)$ with R being $S \to 0S1|\varepsilon$.
- PDA for $L = \{0^k1^k : k \in \mathbb{N}\}$.



- CFG for $L = \{w \in \{0,1\}^* : w = w^R\}$, language of palindromes. Consider $G = (\{S\}, \{0,1\}, R, S)$ with R being $S \to 0S0|1S1|0|1|\varepsilon$.
- PDA for $L = \{w \in \{0,1\}^* : w = w^R\}$, language of palindromes.



- CFG for $L = \{x \in \{(,)\}^* : x \text{ is balanced}\}$. $G = (\{S\}, \{(,)\}, R, S)$ with R being either $S \to (S)|SS|\varepsilon$ or $S \to (S)S|\varepsilon$.
- CFG for $L = \{x \in \{0,1\}^* : x \text{ is not of the form } ww\}$. Consider $G = (\{S, A, B\}, \{0,1\}, R, S)$ with R being $S \to AB|BA|A|B$, and $A \to 0A0|0A1|1A0|1A1|0$, and $B \to 0A0|0A1|1A0|1A1|1$.
- PDA for $L = \{x \in \{0,1\}^* : x \text{ is not of the form } ww\}$.



- $L = \{0^k1^k2^k : k \in \mathbb{N}\}$ is not context-free.

  Proof. Let p be the pumping length and $s = 0^p1^p2^p = uvxyz$. By (3) $|vxy| \leq p$, so it cannot contain all of $0, 1, 2$. Thus $uv^2xy^2z \in L$ must pump one of $0, 1, 2$ less than the others. ☐
- $L = \{ww : w \in \{0,1\}^*\}$ is not context-free.

  Proof. Let p be the pumping length and $s = 0^p1^p0^p1^p = uvxyz$. If vxy is contained in the first half, then $uv^2xy^2z = 0^{p+k}1^{p+f}0^p1^p \in L$ for some $0 < k + f \leq p$. Thus the second half starts with a 1 by the first half starts with a 0. Similarly for if vxy is contained in the second half. If vxy is in both halves, then $uxz = 0^p1^k0^t1^p \in L$ for some $k < p$ and/or $t < p$, either way $uxz \notin L$. ☐
- $L = \{w_1aw_2 : w_1, w_2 \in \{0,1\}^*, \text{and } w_1 \text{ is a substring of } w_2\}$ is not context-free.

  Proof. Let p be the pumping length and $s = 0^p1^pa0^p1^p = uvxyz$. Note we need $a \in x$, so u is a substring of 1^p and v of 0^p. Then $uv^xy^2z = 0^p1^{p+k}a0^{p+\ell}1^p \in L$ with $k > 0$ and/or $\ell > 0$, either way $uv^2xy^2z \notin L$. ☐
- $L = \{0^n1^m : n \leq m^2\}$ is not context-free.

  Proof. Let p be the pumping length and $s = 0^{p^2}1^p = uvxyz$. Let k denote the number of 1's in vy. If $k \geq 1$ then, then $uxz \in L$ but # of 0's in $uxz \geq p^2 - |vy| \geq p^2 - p \geq p(p - k) > (p - k)^2$. If $k = 0$, then $uv^2xy^2z \in L$ has $p^2 + |vy| > p^2$ 0's and p 1's. ☐

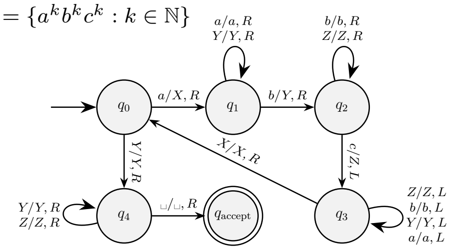
4 Post-TM Definitions

- **TM:** A 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ where Q is a finite set of states, Σ is a finite alphabet, Γ is a finite tape alphabet with $\Sigma \subseteq \Gamma$ and $\sqcup \in \Gamma$, $\delta : Q \times \Sigma \to Q \times \Gamma \times \{L, R\}$ is the transition function, q_0, q_{accept}, q_{reject} are the start, accept, and reject states respectively.

  A TM operates like a PDA, but writing directly to the tape where its input is instead. We assume a TM has a single one-sided infinite tape. A TM M accepts $w = w_1 \cdots w_n \in \Sigma^*$ if there is a computation path that leads from q_0 to q_{accept}.
- **Recognizability:** A language L is recognizable if there is a TM, M, with $\mathcal{L}(M) = L$.
- **Decidability:** A language L is decidable if there is a TM, M, with $\mathcal{L}(M) = L$ and M halts on every input. Such an M is called a decider.
- **TM Variants:** The following are variants of equivalent power to a TM: k-tape TMs, 1-tape two-way infinite TMs, random-access memory (RAM) TM. A non-deterministic TM (NTM), however, is more powerful than a normal TM and functions by letting the transition function not be well-defined. An NTM accepts if any computation path accepts. We often restrict NTMs to have a branching factor of 2, i.e., for any given input the transition function has exactly two possible outputs.
- A language L is recognizable if and only if it is accepted by an NTM.
- If L and $\overline{L}$ are both recognizable then L is decidable. L is decidable if and only if $\overline{L}$ is decidable. If L_1 and L_2 are decidable then so are $L_1 \cup L_2$ and $L_1 \cap L_2$.
- **Strong Church-Turing Thesis:** TMs can model any feasible model of computation with at most polynomial overhead. Thus to show something is recognizable or decidable, we can provide a pseudocode algorithm.
- **Class P:** $P = \bigcup_{k\in\mathbb{N}} DTIME(n^k)$ is the class of languages decidable by a DTM in polynomial time. $DTIME(f(n))$ is the class of languages decidable by a DTM in $O(f(n))$ time.
- **Time Hierarchy:** $DTIME(n^k) \subsetneq DTIME(n^{k+1})$.
- **Efficient UTM:** There is a DTM U such that for any $x \in \{0,1\}^*$ and DTM encoding $\langle M\rangle$, $U(x, \langle M\rangle) = M(x)$. Moreover, if M halts on x in T steps, then U halts on $(x, \langle M\rangle)$ in $O(T \log T)$ steps.
- **Class NP:** A language L is in NP if there is a polynomial $p : \mathbb{N} \to \mathbb{N}$ and poly-time DTM M such that $x \in L$ if and only if there is a $u \in \{0,1\}^{p(|x|)}$ such that $M(x, u) = 1$ for all $x \in \{0,1\}^*$. A langauge $L \in coNP$ if $\overline{L} \in NP$.
- $NP = \bigcup_{k\in\mathbb{N}} NTIME(n^k)$ where $NTIME(f(n))$ is the class of languages decidable by an NTM in $O(f(n))$ time (must be $O(f(n))$ for any branch).
- **Poly-to-One Reductions:** L is poly-to-one reduced to to L', denoted $L \leq_p L'$ if there is a poly-time computable function f such that $x \in L$ if and only if $f(x) \in L'$ for all $x \in \{0,1\}^*$.
- **NP-hard and NP-complete:** L' is NP-hard if for all $L \in NP$, we have $L \leq_p L'$. If $L' \in NP$ also, then L' is NP-complete.
- If L is NP-complete, then $\overline{L}$ is $coNP$-complete.
- **Reduction Properties:** If $L_1 \leq_p L_2$ and $L_2 \leq_p L_3$, then $L_1 \leq_p L_3$. If $L \leq_p L'$ and $L' \in P$, then $L \in P$. If $L \leq_p L'$ and $L' \in NP$ then $L \in NP$ and if L is NP-hard then L' is NP hard.
- **Turing Reductions:** X is Turing reduced to Y, denoted $X \leq_T Y$ if there is a there is an algorithm A that solved Y, and an algorithm B that solves X by calling A.
- $L \leq_T \overline{L}$ for all languages L, but $L \leq_p \overline{L}$ for all languages L if and only if $NP = coNP$. For any NP-complete language L, there is a Turing reduction from the search version of L to the decision version of L. We see this since SAT is NP-complete and $SAT \leq_T$ *Search-SAT*.
- **PTM:** A probabilistic TM (PTM) has a second tape initialized with a random bitstring r. It may then use these random bits to probabilistically find the answer. We say a PTM decides a language L in $T(n)$ time if for every $x \in \{0,1\}^*$, M halts in $T(|x|)$ steps and $P_r(M(x,r) = L(x)) \geq \frac{2}{3}$.
- **Class BPP:** $BPP = \bigcup_{k\in\mathbb{N}} BPTIME(n^k)$ where $BPTIME(f(n))$ is the class of languages decidable by a PTM in $O(f(n))$ time. Alternatively, BPP is the class of languages such that there is a poly-time PTM M and a polynomial $p : \mathbb{N} \to \mathbb{N}$ such that for every $x \in \{0,1\}^*$ we have
$$P_{r \in_R \{0,1\}^{p(|x|)}}\left(M(x,r) = L(x)\right) \geq \frac{2}{3}.$$
- **Class RP:** A language $L \in RP$ if there is a poly-time PTM M such that if $x \in L$ then $P_r(M(x,r) = 1) \geq \frac{2}{3}$ and if $x \notin L$ then $P_r(M(x,r) = 0) = 1$. A language $L \in coRP$ if $\overline{L} \in RP$, i.e., if M is certain when $x \in L$ and probably right when $x \notin L$.
- For any $L \in RP$, there is a PTM M such that if $x \in L$ then $P_r(M(x,r) = 0) \leq (1/3)^{p(|x|)}$ for any polynomial $p : \mathbb{N} \to \mathbb{N}$ (and if $x \notin L$ then $P_r(M(x,r) = 1) = 0$) by running M $p(|x|)$ times.
- For any $L \in BPP$, there is a PTM M such that for all $x \in \{0,1\}^*$ we have $P_r(M(x,r) = L(x)) \geq 1 - 2^{-|x|^d}$ for any $d > 0$. There is also a PTM M such that if $x \in L$ then $P_r(M(x,r) = 1) > \beta + \varepsilon$ and if $x \notin L$ then $P_r(M(x,r) = 1) < \beta - \varepsilon$ for any $\beta > \varepsilon > 0$.
- BPP is subset of non-constructive P.
- **Class ZPP:** $ZPP = \bigcup_{k\in\mathbb{N}} ZTIME(n^k)$ is the class of languages that can be solved in expected polynomial time. $ZTIME(n^k)$ is the class of languages that can be solved in expected time $O(n^k)$. Note $ZPP = RP \cap coRP$.

5 TM Examples

- DTM for $L = \{a^k b^k c^k : k \in \mathbb{N}\}$



- An NTM for the language L of composite numbers could non-deterministically select two numbers $p, q < n$ and check if $pq = n$.
- Since a TM may be represented by a finite bitstring and a language by an infinite bitstring, there are fewer TMs than languages. Thus there non-constructively exist unrecognizable and undecidable languages.
- Self-reject language $SR = \{\langle M\rangle : M \text{ is a TM that doesn't accept } \langle M\rangle\}$ is undecidable.

  Proof. BWOC, let D decide SR. If D accepts $\langle D\rangle$ then $D \notin SR$ so $\mathcal{L}(D) \neq SR$. If D rejects $\langle D\rangle$ then $D \in SR$ so $\mathcal{L}(D) \neq SR$. □
- Self-accept language $SA = \{\langle M\rangle : M \text{ is a TM that accepts } \langle M\rangle\}$ is undecidable.

  Proof. If SA is decidable, then $SR = \overline{SA} \cap \{\langle M\rangle : M \text{ is a TM}\}$ is decidable since $\overline{SA}$ and $\{\langle M\rangle : M \text{ is a TM}\}$ are decidable. □
- Acceptance language $A_{TM} = \{(\langle M\rangle, w) : M \text{ is a TM that accepts } w\}$ is undecidable.

  Proof. BWOC, let D decide A_{TM}. Then we can decide SR by running $D(\langle M\rangle, \langle M\rangle)$ but SR is undecidable. □
- Self-reject language $SR = \{\langle M\rangle : M \text{ is a TM that doesn't accept } \langle M\rangle\}$ is unrecognizable.

  Proof. BWOC, let R recognize SR. If R accepts $\langle R\rangle$ then $R \notin SR$ so $\mathcal{L}(D) \neq SR$. If R rejects $\langle R\rangle$ then $R \in SR$ so $\mathcal{L}(D) \neq SR$. If R loops forever on $\langle R\rangle$ then $R \in SR$ so $\mathcal{L}(D) \neq SR$. □
- The halting problem $A_{halt} = \{(\langle M\rangle, w) : M \text{ is a TM and halts on } w\}$ is recognizable but undecidable.
- Trivial CFG language $ALL_{cfg} = \{G : G \text{ is a CFG and } \mathcal{L}(G) = \Sigma^*\}$ is undecidable.

  Proof. BWOC, let D decide ALL_{cfg}. Construct a PDA $P_{M,w}$ that rejects its input if it is an accepting computation history for $M(w)$ and accepts otherwise. Then run D on the grammar for $P_{M,w}$, if it's true then M rejects w, so we decided A_{TM}. □
- The language $L = \{(G_1, G_2) : G_1, G_2 \text{ are CFGs and } \mathcal{L}(G_1) = \mathcal{L}(G_2)\}$ is recognizable but undecidable.
- The language $L = \{(G, x) : G \text{ is a CFG and } x \in \mathcal{L}(G)\}$ is decidable by simulating G on x.
- The language $L = \{(G, D) : G \text{ is a CFG, } D \text{ is a DFA, and } \mathcal{L}(G) = \mathcal{L}(D)\}$ is undecidable. Otherwise we could decide (G, D_{Σ^*}) to decide ALL_{cfg}.
- Let $L = \{(\langle M_1\rangle, \langle M_2\rangle) : M_1, M_2 \text{ are TMs and } \mathcal{L}(M_1) = \mathcal{L}(M_2)\}$. Then L is doubly unrecognizable.

  Proof. BWOC, let R recognize L. Let M_w be a TM that on any input simulates $M(w)$. Let M_{empty} be a TM that always rejects. Then $(M, w) \in \overline{A_{TM}}$ if and only if $\mathcal{L}(M_w) = \emptyset = \mathcal{L}(M_{empty})$, so $\overline{A_{TM}}$ is recognizable, a contradiction since A_{TM} is recognizable, and thus A_{TM} would be decidable.

  Let $B = \{(\langle M_1\rangle, \langle M_2\rangle) : M_1, M_2 \text{ are TMs and } \mathcal{L}(M_1) \neq \mathcal{L}(M_2)\}$ and let R recognize B (true if and only if $\overline{L}$ is recognizable). Let M_w be as above and M_{all} be a TM that always accepts. Then $(M, w) \in \overline{A_{TM}}$ if and only if $\mathcal{L}(M_w) = \emptyset \neq \mathcal{L}(M_{all})$. □
- The language $L = \{(0, G_1, G_2) : \mathcal{L}(G_1) = \mathcal{L}(G_2)\} \cup \{(1, G_1, G_2) : \mathcal{L}(G_1) \neq \mathcal{L}(G_2)\}$ (over CFGs G_1, G_2) is doubly unrecognizable.
- The language $L = \{(M, j) : M \text{ is a TM that halts on inputs with} \leq j \text{ ones}\}$ is undecidable.

Proof. BWOC, let D decide L. Let $H_{M,x}(w)$ reject if $\text{num}_1(w) \geq 1$, otherwise return $M(x)$. Then $M(x)$ halts if and only if $D(H_{M,x}, 0) = 1$ so we decide the halting problem. □

- The language $L = \{M : M \text{ is a TM and } \forall w \in \{0,1\}^*, M(w0) = M(w1)\}$ is undecidable.

  Proof. BWOC, let D decide L. Let $H_{M,x}(w)$ accept if $w \neq 0$, otherwise return $M(x)$. Then $M(x) = 1$ if and only if $D(H_{M,x}) = 1$ so we decide A_{TM}. □
- The language $L = \{M : M \text{ is a TM and } \forall w \in \{0,1\}^*, M \text{ halts on } w \text{ iff } M \text{ halts on } w^R\}$ is undecidable.

  Proof. BWOC, let D decide L. Let $H_{M,x}(w)$ accept if $w \neq 01$, otherwise return $M(x)$. Then $M(x)$ halts if and only if $D(H_{M,x}) = 1$ so we decide the halting problem. □
- The language $L = \{(G, A) : A \text{ is essential in the CFG } G\}$ is undecidable, where A is an essential variable of a CFG G if for some $w \in \mathcal{L}(G)$, A appears in every derivation of w.

  Proof. For a CFG G, define G' by adding a new variable A with $S \to A$ and $A \to \sigma_1 A \mid \cdots \mid \sigma_n A \mid \varepsilon$ for $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$. Then A is essential in G' if and only if $\mathcal{L}(G) \neq \Sigma^*$, thus we decided ALL_{cfg}. □

6 Complexity Examples

- $L = \{G : G \text{ is a complete graph}\}$ is in P.
- $L = \{n \in \mathbb{N} : n \text{ is prime}\}$ is in P.
- $L = \{(G, x) : G \text{ is a CFG with } x \in \mathcal{L}(G)\}$ is in P.
- $L = \{(G_1, G_2) : G_1 \text{ and } G_2 \text{ are isomorphic graphs}\}$ is in NP with witness given by the graph isomorphism.
- $L = \{G : G \text{ is a graph with a Hamiltonian path}\}$ is in NP with witness given by the Hamiltonian path (a path that visit all vertices).
- $L = SAT = \{\Phi : \Phi \text{ is a satisfiable formula}\}$ is in NP with witness given by the satisfying assignment.
- $L = UNSAT = \{\Phi : \Phi \text{ is an unsatisfiable formula}\}$ is in $coNP$ but not in NP.
- $CLIQUE \leq_p V\text{-}COVER$ where $(G, k) \in CLIQUE$ iff G has a clique (complete subgraph) of size k and $(G, s) \in V\text{-}COVER$ iff G has a vertex cover (set of vertices S.T. every edge has an end in it) of size s.

  Proof. $(G, k) \in CLIQUE \iff (\overline{G}, n-k) \in V\text{-}COVER$ where $\overline{G}$ is the complement (i.e., $E(\overline{G}) = \overline{E(G)}$). This is because if G has a clique of size k, then $\overline{G}$ has a cover of size $n - k$ given by all vertices not in the clique. □
- $SAT \leq_p 3SAT$ where $3SAT$ is SAT but each clause has 3 literals.

  Proof. Given a clause a with one literals, add two new variables p_1, p_2 and add clauses $(a \vee p_1 \vee p_2) \wedge (a \vee p_1 \vee \overline{p_2}) \wedge (a \vee \overline{p_1} \vee p_2) \wedge (a \vee \overline{p_1} \vee \overline{p_2})$. Given a clause $(a \vee b)$ with two literals, add a new variable p and add clauses $(a \vee b \vee p) \wedge (a \vee b \vee \overline{p})$. Given a clause $(z_1 \vee \cdots \vee z_r)$ with r literals, add $r-3$ new variables $y_1, \ldots, y_{r-3}$ and add clauses $(z_1 \vee z_2 \vee y_1) \wedge (z_3 \vee \overline{y_1} \vee y_2) \wedge (z_4 \vee \overline{y_2} \vee y_3) \wedge \cdots \wedge (z_{r-2} \vee \overline{y_{r-4}} \vee y_{r-3}) \wedge (z_{r-1} \vee z_r \vee \overline{y_{r-3}})$. Then the formula is satisfiable if and only if the new formula is. □
- $3SAT \leq_p CLIQUE$.

  Proof. Suppose $\Phi = (x_{1,1} \vee x_{1,2} \vee x_{1,3}) \wedge \cdots \wedge (x_{k,1} \vee x_{k,2} \vee x_{k,3})$. Make a graph G with $V(G) = \{x_{i,j} : i \in \mathbb{Z}_k, j \in \mathbb{Z}_3\}$ and with an edge between $x_{i,j}$ and $x_{i',j'}$ if and only if $i \neq i'$ and $x_{i,j} \neq \overline{x_{i',j'}}$. Then Φ is satisfiable if and only if G has a clique of size k (the clique would provide a satisfying assignment since it selects one true literal from each clause). □
- SAT is NP-complete.

  Proof. Let L be an NP language. Let M be a TM with $Q = \{q_0, \ldots, q_w\}$ where $q_0 = q_{\text{start}}$ and $q_w = q_{\text{accept}}$ and $\Gamma = \{0, 1, \sqcup\}$. Suppose M runs in $p(n)$ steps and has witness of length $f(n)$ for p, f polynomials $\mathbb{N} \to \mathbb{N}$. We create a formula to check if M is a valid TM accepting x, it is satisfiable if and only if $x \in \mathcal{L}(M)$.

  - Add variables $y_{i,j}$ for $1 \leq i \leq p(n)$ and $0 \leq j \leq w$ denoting at time i, M is in state q_j.
  - Add variables $h_{i,j}$ for $1 \leq i \leq p(n)$ and $0 \leq j \leq p(n)$ denoting at time i, the head is at cell j.
  - Add variables $r_{i,j,k}$ for $1 \leq i, j \leq p(n)$ and $k \in \{0, 1, \sqcup\}$ denoting at time i, cell j contains symbol k.
  - (G_1) Add clauses $y_{i,0} \vee \cdots \vee y_{i,w}$ for all $1 \leq i \leq p(n)$ and $y_{i,j} \implies \overline{y_{i,j'}}$ for all $1 \leq i \leq p(n)$ and $1 \leq j, j' \leq w$ with $j \neq j'$. That is, M is in exactly one state.
  - (G_2) Add clauses $h_{i,0} \vee \cdots \vee y_{i,p(n)}$ for all $1 \leq i \leq p(n)$ and $h_{i,j} \implies \overline{h_{i,j'}}$ for all $1 \leq i \leq p(n)$ and $1 \leq j, j' \leq p(n)$ with $j \neq j'$. That is, M's head is at exactly one cell.
  - (G_3) Add clauses $r_{i,j,0} \vee r_{i,j,1} \vee r_{i,j,\sqcup}$ for all $1 \leq i, j \leq p(n)$ and $r_{i,j,k} \implies \overline{h_{i,j,k'}}$ for all $1 \leq i, j \leq p(n)$ and $k, k' \in \{0, 1, \sqcup\}$ with $j \neq j'$. That is, M's tape has exactly one symbol in each cell.
  - (G_4) Add clauses $y_{1,0}$ (initial state) and $h_{1,0}$ (initial head) and $r_{1,0,x_1} \wedge \cdots \wedge r_{1,n-1,x_n}$ (input) and $\overline{r_{1,n,\sqcup}} \wedge \cdots \wedge \overline{r_{1,n+f(n)-1,\sqcup}}$ (witness) and $r_{1,n+f(n),\sqcup} \wedge \cdots \wedge r_{1,p(n),\sqcup}$ (blank tape). That is, M is initially configured correctly.
  - (G_5) Add clause $y_{p(n),w}$. That is, M accepts.
  - (G_6) Add clauses $\overline{h_{i,j}} \wedge r_{i,j,k} \implies r_{i+1,j,k}$ for all i, j and $k \in \{0, 1, \sqcup\}$ (unchanged cells) and if $\delta(q_m, k) = (q_{m'}, k', R)$ then for all i, j add $h_{i,j} \wedge y_{i,m} \wedge r_{i,j,k} \implies y_{i+1,m'}$ (state) and $h_{i,j} \wedge y_{i,m} \wedge r_{i,j,k} \implies h_{i+1,j+1}$ (head, do $j - 1$ for left) and $h_{i,j} \wedge y_{i,m} \wedge r_{i,j,k} \implies r_{i+1,j,k'}$ (content). That is, M follows its transition rules. □
- $SAT \leq_T Search\text{-}SAT$.

  Proof. Suppose Φ is our formula with variables $x_1, \ldots, x_n$. Set $x_1 = 1$ and see if the resulting formula is satisfiable. If so $x_1 = 1$ in our assignment, otherwise $x_1 = 0$. Expand our assignment by repeating with $x_2 = 1$, so on so forth. □
- $CLIQUE \leq_T Search\text{-}CLIQUE$.

  Proof. Suppose G is our graph and k is given. Pick a node $x \in V(G)$ with $x \notin C$. If $G - x$ has a clique of size k, set $G = G - x$, otherwise add x to C. Repeat until $|C| = k$. □
- $L = \{f \in \mathbb{Z}_p[x] : f = 0\}$ is in BPP. We randomly pick $a \in \mathbb{Z}_p$ and return 1 iff $p(a) = 0$. Then $P(A(f) \text{ is incorrect}) \leq \frac{d}{p}$ where d is the degree of f. By repeating this we can reduce the error.
- $L = \{(A, B, C) \in (\mathbb{R}^{n \times n})^3 : AB = C\}$ is in $coRP$. Randomly select a column vector $x \in \mathbb{R}^n$ and check if $ABx = Cx$. If $AB = C$, then returns true with probability 1, if $AB \neq C$, then returns false with probability at least $\frac{1}{2}$.