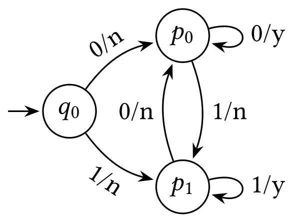
6 Automata Theory Cheatsheet

* **Alphabet** is a finite set of symbols, commonly denoted Σ.
* **Word** $w \in \Sigma^*$ is a finite sequence of symbols from alphabet Σ For example, $w = abacaba \in \{a, b, c\}^*$.
* **Length** of a word: $|w| = n$, where n is the number of symbols in word w. For example, $|abacaba| = 7$.
* **Empty word** ε is a word of length 0.
* **Concatenation** of words w_1 and w_2 is $w_1 \cdot w_2 = w_1 w_2$.
* **Power** of a word w is $w^n = w \cdot w \cdot \ldots \cdot w$ (n times).
* **Reverse** of a word w is w^R.
* **Language** L over an alphabet Σ is a set of words $L \subseteq \Sigma^*$.
* **Empty language** $\varnothing$ is a language that contains no words.
* **Singleton language** $\{w\}$ is a language that contains only one word w.
* **Empty string language** $\{\varepsilon\}$ is a language that contains only one empty word ε.
* **Operations** on languages:
  - **Union**: $L_1 \cup L_2 = \{w \mid w \in L_1 \lor w \in L_2\}$
  - **Intersection**: $L_1 \cap L_2 = \{w \mid w \in L_1 \land w \in L_2\}$
  - **Complement**: $\neg L = \{w \mid w \notin L\}$
  - **Concatenation**: $L_1 \cdot L_2 = \{ab \mid a \in L_1, b \in L_2\}$
  - **Kleene star (Kleene closure)**: $L^* = \bigcup_{k=0}^{\infty} \Sigma^k$
* **Equivalence**: $L_1 \equiv L_2 \leftrightarrow (L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) = \varnothing$
* **Regular language** is a language that can be defined by a regular expression.
  Regular languages are defined inductively (recursively):
  - The empty language $\varnothing$ is regular.
  - For any $a \in \Sigma$, the singleton language $\{a\}$ is regular.
  - If A is a regular language, then A^* (Kleene star) is also regular.
  - If A and B are regular languages, then $A \cup B$ (union) is also regular.
  - If A and B are regular languages, then $A \cdot B$ (concatenation) is also regular.
  - No other languages over Σ are regular.
* **REG (set of regular languages)** is set over an alphabet Σ
  $\mathrm{REG} = \bigcup_{k=0}^{\infty} \mathrm{Reg}_k = \mathrm{Reg}_\infty$.
  - $\mathrm{Reg}_0 = \{\varnothing, \{\varepsilon\}\} \cup \{\{c\} \mid c \in \Sigma\}$.
  - $\mathrm{Reg}_{i+1} = \mathrm{Reg}_i \cup \{A \cdot B, A \cup B \mid A, B \in \mathrm{Reg}_i\} \cup \{A^* \mid A \in \mathrm{Reg}_i\}$.
* REG is closed under union, concatenation, and Kleene star operations.
* **Regular expressions (regex)** is a sequence of special characters that define a regular language or an operation over regular languages. The table below illustrates the correspondence between regular languages and regular expressions. Here, $c \in \Sigma$ denotes the symbol of a given alphabet, $A \subseteq \Sigma^*$ and $B \subseteq \Sigma^*$ are some regular languages, α and β are regular expressions. In regular expressions, concatenation is denoted by $\cdot$ (can be omitted in regex), union by |, Kleene star by $*$, and the grouping is made by parentheses.

Language	Regex
$\varnothing$	$\varnothing$
$\{\varepsilon\}$	ε
$\{c\}$	c
$A \cup B$	$\alpha \vert \beta$
$A \cdot B$	$\alpha\beta$
A^*	α^*
$A \cdot A^*$	α^+
$A \cup \{\varepsilon\}$	$\alpha?$

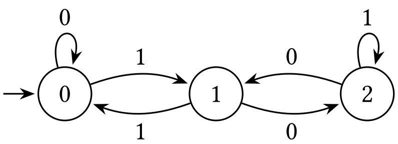
* **Deterministic Finite Automaton (DFA)** is a 5-tuple $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$, where:
  - Σ is an alphabet;
  - $Q = \{q_1, \ldots, q_n\}$ is a finite set of states;
  - $q_0 \in Q$ is an initial state;
  - $F \subseteq Q$ is a set of final (terminal, accepting) states;
  - $\delta \colon Q \times \Sigma \to Q$ is a transition function.

- Language **accepted** by an automaton $\mathcal{A}$ is the set $L(\mathcal{A}) = \{w \mid \delta(q_0, w) \in F\}$.
- **Nondeterministic Finite Automaton (NFA)** is 5-tuple $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$, where:
  - Σ is an alphabet;
  - $Q = \{q_1, \dots, q_n\}$ is a finite set of states;
  - $q_0 \in Q$ is an initial state;
  - $F \subseteq Q$ is a set of final (terminal, accepting) states;
  - $\delta : Q \times \Sigma \to 2^Q$ is a transition function.
- **NFA to DFA** conversion algorithm:
  1. Set initial state of NFA to initial state of DFA.
  2. Take the states in the DFA, and compute in the NFA what the union δ of those states for each letter in the alphabet and add them as new states in the DFA.
  3. Set every DFA state as accepting if it contains an accepting state from the NFA
- **Epsilon-NFA (ε-NFA)** is a NFA that allows ε-moves, that is, the automaton can change state without consuming input.
  - $\delta\colon Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$.
- **ε-NFA to NFA**:
  1. Find transitive-closure of ε.
  2. Back-propagate accepting states over ε-transitions.
  3. Perform symbol-transition back-closure over ε-transitions.
  4. Remove ε-transitions.
- **Pumping lemma** states that if L is a regular language, then there exists an integer $n > 1$ depending only on L, such that $\forall w \in L$, $|w| > n$ can be written as $w = xyz$, such that:
  1. $|y| > 0$, i.e. $y \neq \varepsilon$
  2. $|xy| \leq n$
  3. $\forall k \geq 0$, word xy^kz is also in language L
- **Mealy**[1] **machine** is a finite-state machine whose output is determined both by the current state and the current input.
  Formally, $\mathcal{M}_{\text{Mealy}} = \{\Sigma, \Omega, Q, q_0, \delta, \lambda_{\text{Mealy}}\}$, where:
  - Σ is an input alphabet;
  - Ω is an output alphabet;
  - $Q = \{q_1, \dots, q_n\}$ is finite set of states;
  - $q_0 \in Q$ is an initial state;
  - $\delta\colon Q \times \Sigma \to Q$ is a transition function;
  - $\lambda_{\text{Mealy}}\colon Q \times \Sigma \to \Omega$ is an output function.



This Mealy machine's output is "y" whenever the last two symbols in the input are the same, and "n" otherwise.

- **Moore**[2] **machine** is a finite-state machine whose output is determined only by the current state.
  Formally, $\mathcal{M}_{\text{Moore}} = (\Sigma, \Omega, Q, q_0, \delta, \lambda_{\text{Moore}})$, where:
  - Σ is an input alphabet;
  - Ω is an output alphabet;
  - $Q = \{q_1, \dots, q_n\}$ is a finite set of states;
  - $q_0 \in Q$ is an initial state;
  - $\delta\colon Q \times \Sigma \to Q$ is a transition function;
  - $\lambda_{\text{Moore}}\colon Q \to \Omega$ is an output function.



This Moore machine's output is modulo 3 of a binary number.

- **Emptiness**. Language $L(M)$ is not empty ($L \neq \varnothing$) if M accepts a word w such that $|w| \leq n$.
- **Infiniteness**. Language $L(M)$ is infinite ($|L| = \infty$) if M accepts a word w such that $n \leq |w| < 2n$.
- **Myhill-Nerode theorem** states that the following three statement are equivalent:
  1. $L \subseteq \Sigma^*$ is accepted by some finite automaton (L is regular).
  2. L is the union of some equivalence classes of right invariant equivalence relation of finite index.
  3. Let R_L be a relation over words: $x\ R_L\ y$ iff $\forall z \in \Sigma : xz \in L \equiv yz \in L$. Then the quotient $\Sigma^*/_{R_L}$ is finite.

[1] Mealy, George H. (1955). A Method for Synthesizing Sequential Circuits. *The Bell System Technical Journal*, 34(5), 1045–79.
[2] Moore, Edward F. (1956). Gedanken-Experiments on Sequential Machines. *Automata Studies, Annals of Mathematical Studies* (34), 129–153.

- **Formal grammar** is 4-tuple $\mathcal{G} = (V, T, S, \mathcal{P})$, where:
  - $\mathcal{V}$ is a vocabulary, a set of variables or non-terminal symbols.
  - T is a set of terminal symbols disjoint from $\mathcal{V}$.
  - S is a start symbol, also called sentence symbol.
  - $\mathcal{P}$ is a set of production rules, each rule of the form: $\mathcal{V}^* S \mathcal{V}^* \rightarrow \mathcal{V}^*$.
- Binary relation $\Rightarrow$ over an grammar $\mathcal{G}$ is defined by:
  $x \Rightarrow y \iff \exists u, v, p, q \in \mathcal{V} : (x = upv) \land (p \rightarrow q \in \mathcal{P}) \land (y = uqv)$.
  Pronounce as "y is directly derivable from x".
- Binary relation $\Rightarrow^*$ over a grammar $\mathcal{G}$ is defined as reflexive transitive closure of $\Rightarrow$.
  Pronounced as "y is derivable from x".
- **Backus-Naur Form (BNF)** is notation to describe the syntax of formal language. A BNF specification is a set of derivation rules, written as follows:

  $\langle symbol \rangle \quad ::= \quad \langle expression \rangle$

  where:
  - $\langle symbol \rangle$ is a non-terminal symbol that is enclosed in angle brackets.
  - $\langle expression \rangle$ consists of one or more sequences of either terminal or non-terminal symbols where each sequence is separated by a vertical bar indicating a choice.
  - ::= is a symbol that separates the production rule for a non-terminal symbol.