Example

$L = \{a^n b^n \ : \ n \in \omega\}$ is not regular.

Definition (Context-free language)

Let V be a finite set of nonterminal characters, Σ be a finite set of terminal characters, and $S \in V$ be the start symbol. The empty word λ is considered as a terminal. Context-free languages are generated from the start symbol S by finitely many production rules of the form $A \to \alpha$, where A is a single nonterminal symbol, and α is a non-empty word of terminal and/or nonterminal symbols.

Several production rules with the same nonterminal on the left hand side are merged into a single rule separating the right hand sides by $|$.

Example

With $\Sigma = \{a, b\}$ the rules $S \to aSa$, $S \to bSb$, $S \to \lambda$ generate the word $aabbaa$ as follows:

$$S \to aSa \to aaSaa \to aabSbaa \to aabbaa$$

Example

$\{a^n b^n : n \geq 0\}$ is context-free.

Proof.

$$S \to aSb, \quad S \to \lambda$$

Examples

- $L = \{w \,:\, w \in \{(,)\}^*, w \text{ has matching parentheses}\}$

$$S \to SS, \quad S \to (S), \quad S \to ().$$

- $L = \{ww^R \,:\, w \in \{a,b\}^*\}$.

$$S \to aSa, \quad S \to bSb, \quad S \to \lambda.$$

- $\{b^n a^m b^{2n} \,:\, n \geq 0,\ m \geq 0\}$.

$$S \to bSbb \,|\, A, \quad A \to aA \,|\, \lambda.$$

- $L = \{w \,:\, w \in \{a,b\}^*$, the # of a's and the # of b's in w are different$\}$.

$$S \to T \,|\, U, \quad T \to VaT \,|\, VaV \,|\, TaV$$

$$U \to VbU \,|\, VbV \,|\, UbV, \quad V \to aVbV \,|\, bVaV \,|\, \lambda$$

Theorem

All regular languages are context-free.

Proof.

A regular grammar is context-free. Other way: Regular expressions can easily be converted into context-free production rules. We only hint some examples: $S \to Sa \mid \lambda$ generates a^*; $S \to Ab$, $A \to Aa \mid \lambda$ generates a^*b. $a(b|c)^*$ can be produced by $S \to aX$, $X \to bX \mid cX \mid \lambda$; etc. □

Definition (Recall: Regular languages)

The family of regular languages over the alphabet Σ is defined recursively as follows.

- $\emptyset$ and $\{\lambda\}$ are regular.
- For each $a \in \Sigma$, $\{a\}$ is regular.
- If A, B are regular, then so are $A|B$, AB, and A^*.

Theorem

Context-free languages are closed under union, concatenation, and Kleene-star.

Proof.

Suppose the two context-free languages are represented by production rules with starting symbols S_1 and S_2, respectively. To get the union add the rule $S \to S_1 \,|\, S_2$. To get concatenation add the rule $S \to S_1 S_2$. Finally, to get the Kleene-star, add the rule $S \to S_1 S \,|\, \lambda$. □

Theorem (Pumping lemma)

For every context-free language L there is an integer $p \geq 1$ such that every word $s \in L$ of length at least p can be written in the form $s = uvwxy$ such that $|vx| \geq 1$, $|vwx| \leq p$, and $uv^n wx^n y \in L$ for all $n \geq 0$.

Example

$L = \{a^n b^n c^n \ : \ n \geq 0\}$ is not context-free.

Proof.

By way of contradiction, assume L is context-free and let p be the constant from the pumping lemma. Each word $a^n b^n c^n$ can be written as $uvwxy$ with $|vx| \geq 1$, $|vwx| \leq p$ so that $uv^2 wx^2 y$ must belong to L. Nevertheless, for large enough n it is not of the form $a^k b^k c^k$. □

Theorem

the class of context-free languages is

- not closed under intersection,
- not closed under complementation.

Proof.

a) Both $L_1 = \{a^m b^m c^n \ : \ m, n \geq 0\}$ and $L_2 = \{a^m b^n c^n \ : \ m, n \geq 0\}$ are context-free, however $L_1 \cap L_2 = \{a^n b^n c^n : n \geq 0\}$ is not.
(b) The class of context-free languages is closed under taking union. As $A \cap B = (A' \cup B')'$, the same class cannot be closed under complementation. □

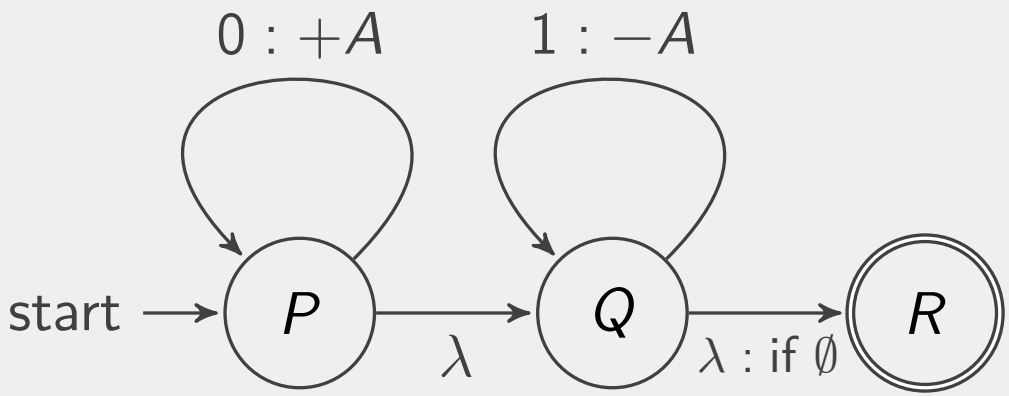
Pushdown automata

A finite-state machine just looks at the input signal and the current state: it has no stack to work with, and therefore is unable to access previous values of the input. It can only choose a new state, the result of following the transition. A pushdown automaton (PDA) differs from a finite state machine in two ways

- It can use the top of the stack to decide which transition to take.
- It can add/remove from the stack as part of a transition.

Given an input symbol, current state, and stack symbol, the automaton can follow a transition to another state, and optionally push or pop from the stack.

PDA which recognizes the language $\{0^n 1^n : \ n \geq 0\}$



- in state P any time the symbol 0 is read, one A is pushed onto the stack.
- at any moment the automaton may move from state P to state Q.
- in state Q, for each symbol 1 read, one A is popped.
- Finally, the machine may move from state Q to accepting state R only when the stack is empty.

Theorem

A language is context-free if and only if it can be accepted by a pushdown automaton.

Definition (Context-sensitive grammar)

A formal grammar in which the left-hand sides and right-hand sides of any production rules may be surrounded by a context of terminal and nonterminal symbols: Each rule is of the form

$$S \to \lambda \qquad \alpha A \beta \to \alpha \gamma \beta$$

where A is a nonterminal symbol, and α, β, γ are non-empty words of terminal and/or nonterminal symbols.

The word γ is not allowed to be empty!

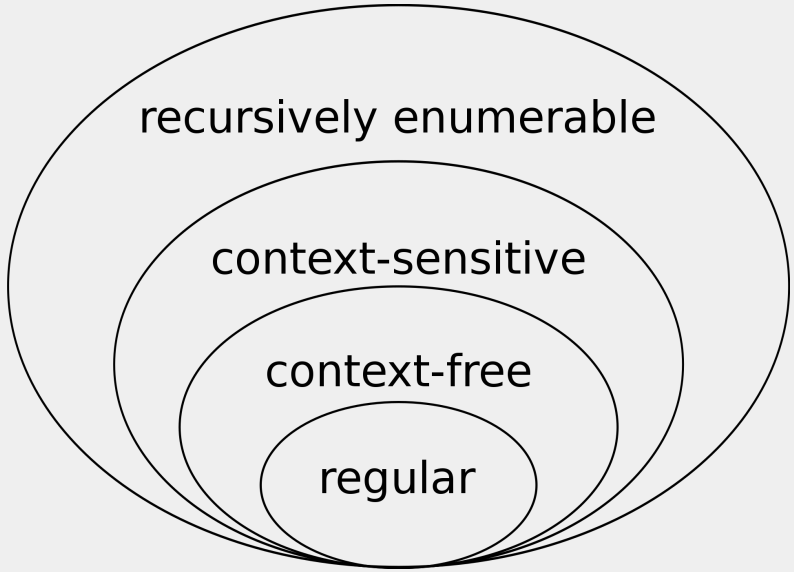
Example

The following languages can be generated by context-sensitive grammars:

- $L = \{a^n b^n c^n \ : \ n \geq 0\}$.
- $L = \{a^n b^n c^n d^n \ : \ n \geq 0\}$
- $\vdots$
- $L = \{a^n b^m c^n d^m \ : \ n, m \geq 0\}$.
- $\vdots$
- It has been shown that almost all natural languages can be characterized by context-sensitive grammars

Definition (Unrestricted grammar)

This is the most general class of grammars in the Chomsky hierarchy. No restrictions are made on the productions of an unrestricted grammar, other than each of their left-hand sides being non-empty. This grammar class can generate arbitrary recursively enumerable languages.

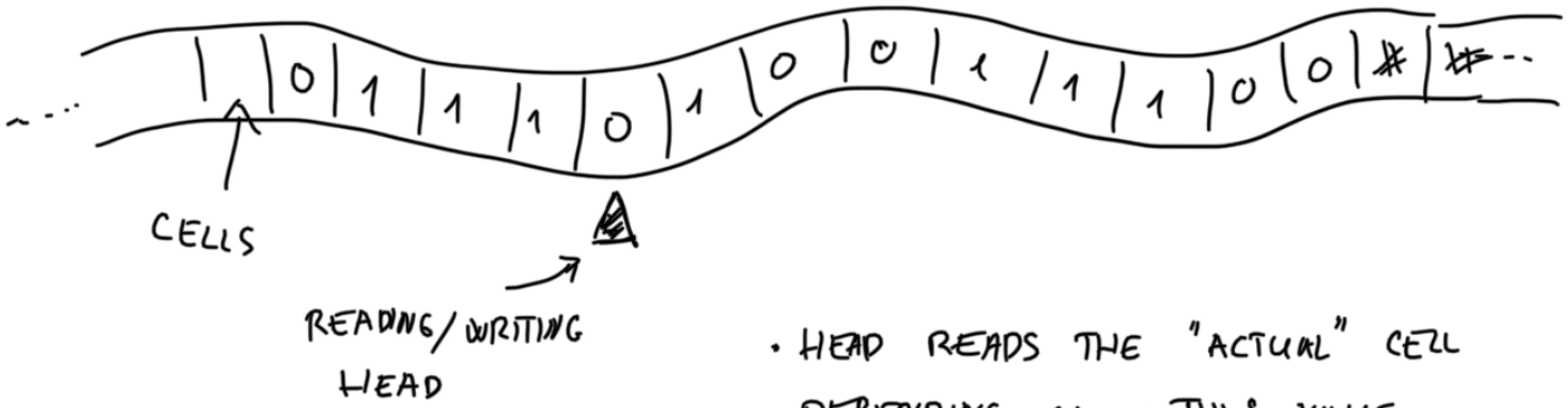


Machine description?

A Turing machine is similar to a finite state automaton. It has two parts: $k \geq 1$ tapes and a control unit. Each tape is a one-way infinite sequence of cells and each cell contains a letter from a finite alphabet Σ. Except for finitely many cells, all the cells are filled in with a special *blank* symbol #. The control unit is similar to a finite state automata in that it has a finite set Q of states with two distinguished states, START and STOP. Over each tape there is a reading / writing head and there are three transitions functions α, β and γ. A computation is performed as follows. Initially the machine is in START state, the heads are on the first cells. All cells are blank except for the beginning of the tapes where the input strings are written. The machine performs steps: at each step the control unit is at some state and the heads are over certain cells. Depending on this state and the contents of the cells at the position where the heads are, the transition function α specifies the next state; β specifies the symbols to be written at the position of the heads; and γ tells how each heads move: once cell ahead or one cell back. The machine halts if it reaches the STOP state.

TURING MACHINES

TAPE:



- HEAD READS THE "ACTUAL" CELL DEPENDING ON
  - THIS VALUE
  - THE STATE OF THE MACHINE

→ IT WILL WRITE SOME LETTER TO THE CELL

→ MOVE THE HEAD TO THE LEFT OR RIGHT OR STAY

↔ CHANGE THE STATE OF THE MACHINE

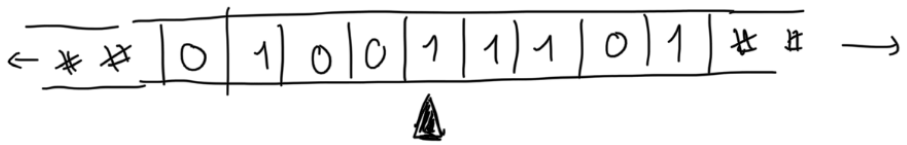
Definition (Turing machines)

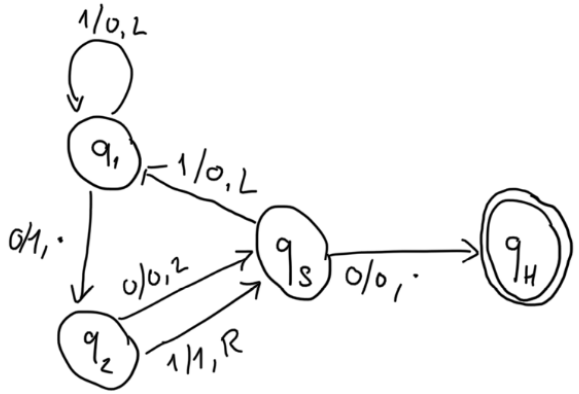
A Turing machine is a tuple $T = \langle k, \Sigma, Q, \alpha, \beta, \gamma \rangle$, where $k \geq 0$ is the number of tapes, Σ is a finite set of alphabet, the special blank symbol $* \in \Sigma$, Q is a finite set of states with the distinguished states START and STOP in it, and

$$\begin{aligned} \alpha &: Q \times (\Sigma \cup \{\#\})^k \to Q \\ \beta &: Q \times (\Sigma \cup \{\#\})^k \to (\Sigma \cup \{\#\})^k \\ \gamma &: Q \times (\Sigma \cup \{\#\})^k \to \{-1, 0, 1\} \end{aligned}$$

are arbitrary functions. The set of k-tape Turing machines over the alphabet Σ is denoted by TM^k_Σ.

EXAMPLE: 1-TAPE TM OVER $\Sigma = \{0,1\}$





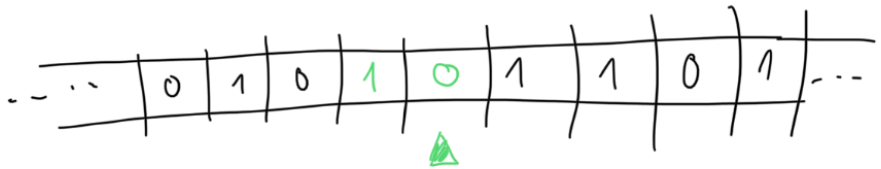
q_S = START STATE

q_H = STOP STATE

0/1, L — IF THE CONTENT OF THE CELL IS 0, THEN MODIFY IT TO 1 AND MOVE THE HEAD TO THE LEFT

1/1, R — FROM 1 IT MAKES 1 MOVES TO THE RIGHT

1/1, · — · = STAY WITH THE HEAD



q_S , q_1 , q_2 , q_S , q_H END OF COMPUTATION

A Turing machine can be regarded as a partial function:

- Input = words written at the beginning of the tapes.
- After performing computation on this input,
  - if the machine reaches the STOP state, then the output is the string in the first tape (the contents of the cells of the first tape from the initial cell until the first cell that contains #).
  - the machine may never reach the STOP state for some input $\Longrightarrow$ the machine does not halt.