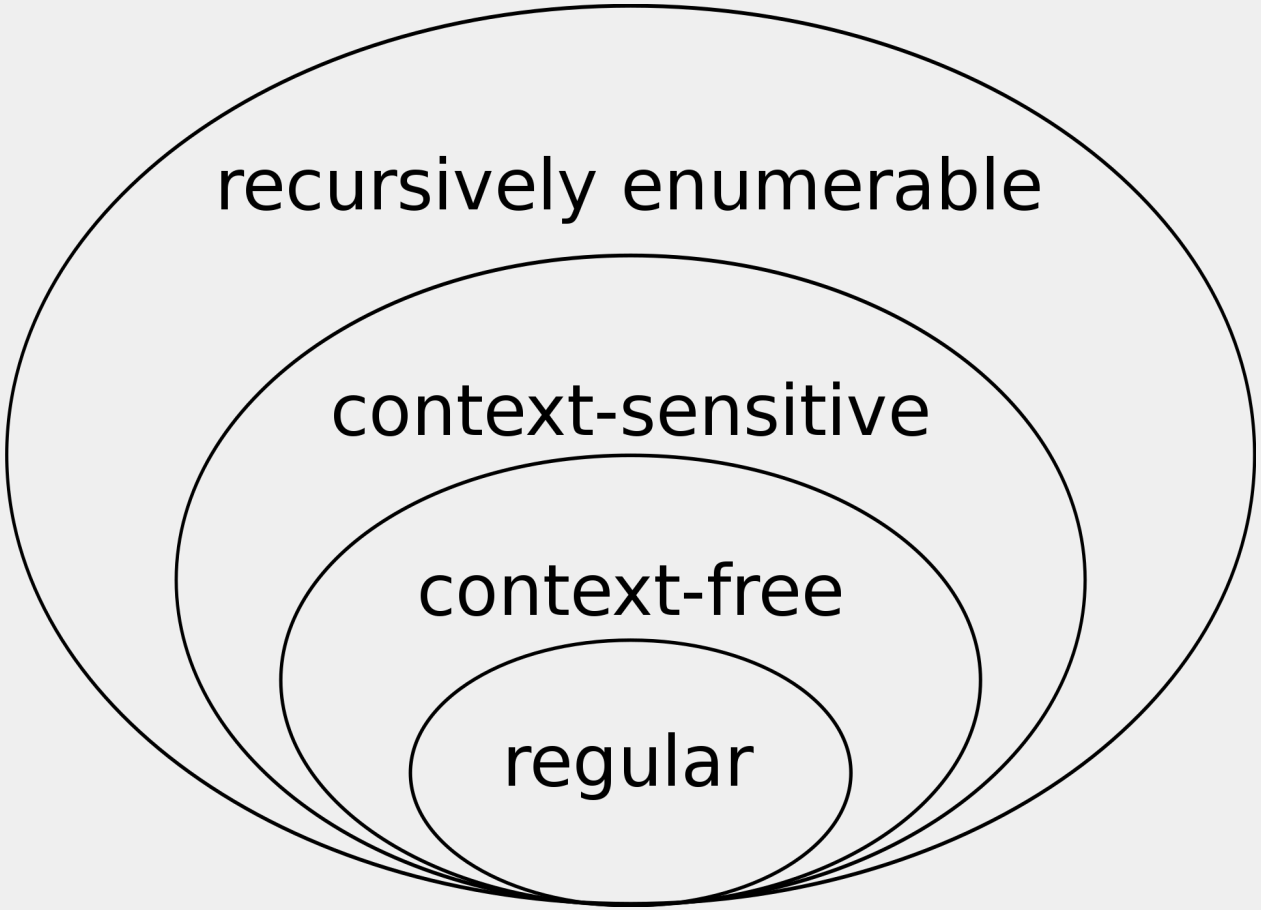
Languages we consider:

- generated by some formal grammar
- described or matched by a particular expression
- accepted by some algorithm (finite-state automaton, Turing machine)

Applications: Logic, Computer science, Linguistics, ...

> *A formal grammar describes how to form strings from a language's vocabulary (or alphabet) that are valid according to the language's syntax. Linguist Noam Chomsky theorized that four different classes of formal grammars existed that could generate increasingly complex languages. Each class can also completely generate the language of all inferior classes.*

Chomsky's hierarchy:



Definition (Operations on languages)

On languages over the alphabet Σ the usual operators are the following.

- Union: $L_1 \cup L_2$ is the set theoretical union of L_1 and L_2. Frequently this operation is written as $L_1|L_2$.
- Concatenation: $L_1 L_2 = \{w_1 w_2 : w_1 \in L_1, w_2 \in L_2\}$.
- Kleene star: $L^* = \{\lambda\} \cup L \cup L^2 \cup L^3 \cup \ldots$.

Definition (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^*.

Generation of regular languages are described by *regular expressions*.

Definition (Regular expressions)

Given a finite alphabet Σ we define the set of regular expressions as follows.

- $\emptyset$, λ and a for each $a \in \Sigma$ are regular expressions.
- If σ and τ are regular expressions the so are $(\sigma|\tau)$, $(\sigma\tau)$, and $(\sigma)^*$.

Whenever it is clear we omit the parentheses. For example, $a|b^*$ is the language

$$\{a, \lambda, b, bb, bbb, \ldots\},$$

while $(a|b)^*$ is the set of all words over the alphabet $\{a, b\}$.

Examples

Consider the following languages over the alphabet $\Sigma = \{a, b\}$.

- The set of strings that have at least one b.

$$a^* b (a|b)^*$$

- The set of strings that have at most one b.

$$a^* | a^* b a^*$$

- The set of strings that end in 3 consecutive b's.

$$(a|b)^* bbb$$

Theorem

- Every finite language L is generated by some *DFA*.
- Every finite language is regular.

Proof.

(a) Let $L = \{w^i : i < N\}$ be a finite language and suppose $w^i = x_1^i x_2^i \cdots x_{n_i}^i$. We describe a *DFA* that accepts L. For each $i < N$ and $j < n_i$ let q_j^i be the states and let q_λ be the starting state. Halting states are the $q_{n_i}^i$, and transitions are $\delta(q_j^i, x_{j+1}^i) = q_{j+1}^i$ (for $j < n_i$) and $\delta(q_\lambda, x_1^i) = q_1^i$. (The states $q_j^{i_1}$ and $q_j^{i_2}$ are the same if the first j symbols of w_{i_1} and if w_{i_2} are the same.)
(b) Every finite set is the finite union of its singletons, all of which are regular by definition. □

Generative grammars

A grammar consists of a set of production rules, rewriting rules for transforming strings. Each rule specifies a replacement of a particular string (its left-hand side) with another (its right-hand side). A rule can be applied to each string that contains its left-hand side and produces a string in which an occurrence of that left-hand side has been replaced with its right-hand side.

Example

$S \rightarrow aSb$
$S \rightarrow ba$

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aababb$$

The language generated by this grammar is

$$\{a^n bab^n : \ n \geq 0\} = \{ba, abab, aababb, aaababbb, \ldots\}$$

This grammar is context free: the production rules can be applied to a nonterminal symbol regardless of its context.

Example

$S \rightarrow a$
$S \rightarrow SS$
$aSb \rightarrow b$

This grammar is context sensitive because production rule #3 can be applied depending on the context.

HW: what is the language generated by this grammar?

Example

$S \rightarrow aS, \quad S \rightarrow bS, \quad S \rightarrow a, \quad S \rightarrow b.$

HW: This grammar is context-free and generates the same language.

Regular grammar = Type-3 grammar

Definition (Regular grammar via an example)

- All production rules have at most one non-terminal symbol, and
- that symbol is either always at the end or always at the start of the rule's right-hand side.

$$\begin{aligned} A &\to a \\ A &\to aB \\ A &\to \lambda \end{aligned}$$

here A, B, S are non-terminal symbols, $a \in \Sigma$ is a terminal symbol and λ is the empty symbol.

NOTE: $S \to aT$, $T \to Sb$, $S \to \lambda$ is not regular!
Examples:

- $S \to aS$, $S \to bA$, $A \to \lambda$, $A \to cA$ $\qquad (a^*bc^*)$
- $S \to A$, $A \to aA$, $A \to B$, $B \to bC$, $C \to \lambda$, $C \to cC$

Theorem (Kleene)

For a language the following are equivalent

- Generated by a regular grammar.
- Generated by a regular expression (=regular language)
- Accepted by a *DFA*.

Are there non-regular languages?

Example

$L = \{a^n b^n : n \in \omega\}$ is not accepted by any *DFA*.

The proof is based on the pumping lemma:

Theorem (Pumping lemma)

If the language L is generated by a *DFA*, then there exist $p \geq 0$ (depending on L only) such that for all $uwv \in L$ with $|w| \geq p$, there exists a partition $w = xyz$ such that $|xy| \leq p$, $|y| > 0$ and for all $i \geq 0$, $uxy^i zv \in L$. Here u and v might be empty.

uxyzv
uxyyzv
uxyyyzv
uxyyyyzv
uxyyyyyzv
uxyyyyyyzv
uxyyyyyyyzv

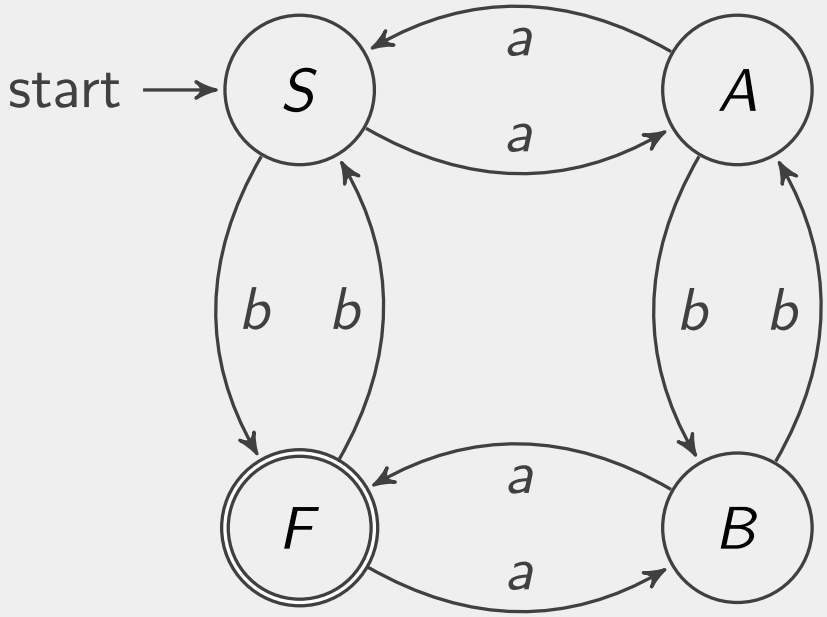
Example

$L = \{a^n b^n : n \in \omega\}$ is not accepted by any *DFA*.

Proof.

If L is accepted by a *DFA*, then there is p such that whenever $|a^n b^n| \geq p$ then $a^n b^n$ can be written as xyz with y not empty, $|xy| < p$ and such that for all $i \geq 0$, $xy^i z \in L$. But clearly there is i so that $xy^i z$ is not of the form $a^\ell b^\ell$ for some i. □

Homework



What is the language L generated by this DFA?
HW***: Give a regular expression for L.