Pre-lecture brain teaser

Is the following language regular? Either way, prove it.

$L = \{\text{strings of properly matched open and closing parentheses}\}$

Ex: { ()(), ((()())))(), ... }

CS/ECE-374: Lecture 8 - Context-Free languages and Turing Machines

Lecturer: Nickvash Kani
Chat moderator: Samir Khan

February 16, 2021

University of Illinois at Urbana-Champaign

Pre-lecture brain teaser

Is the following language regular? Either way, prove it.

$L = \{\text{strings of properly matched open and closing parentheses}\}$

Not regular

$L = \{\varepsilon, (), (()), ((())), (((()))), \ldots$

$\equiv 0^n 1^n$

Prove via Fooling Set

$F = \{ "(" ^n \mid n \geq 0 \}$

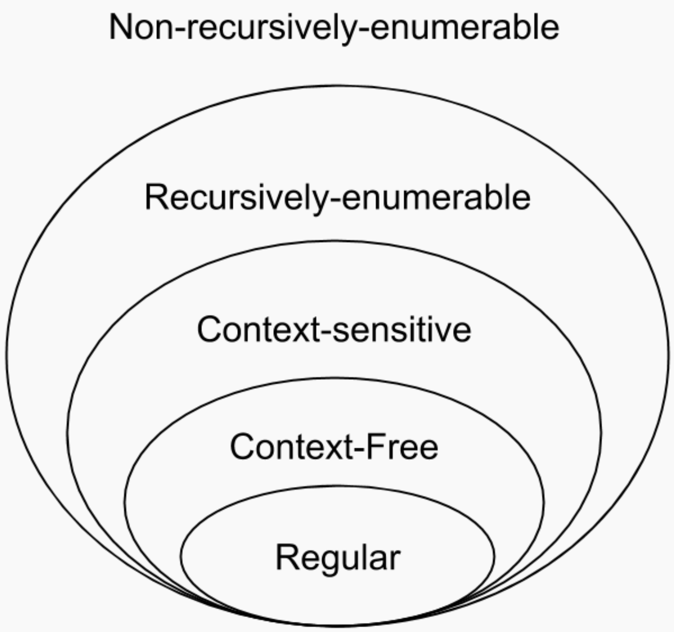
$(^i$ & $(^j$ exist a $w =)^i$ where $(^i)^i \in L$

$(^j)^i \notin L$

Since F is infinite, DFA M representing L is also ~~[illegible]~~ infinite.

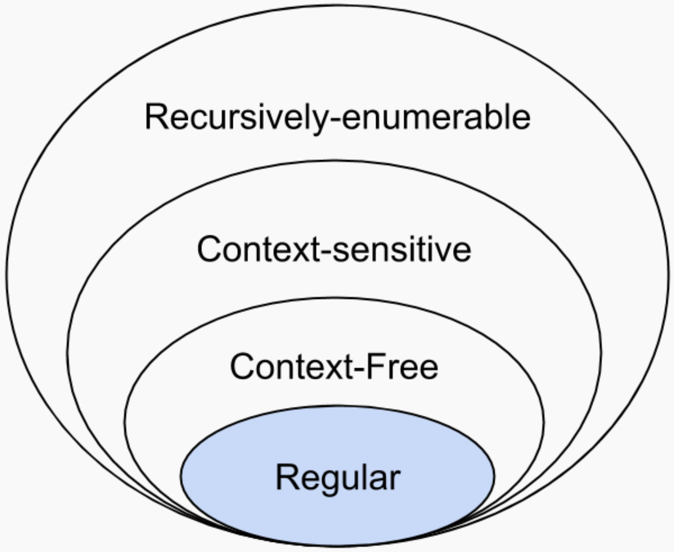
Larger world of languages!

Chomsky Hierarchy



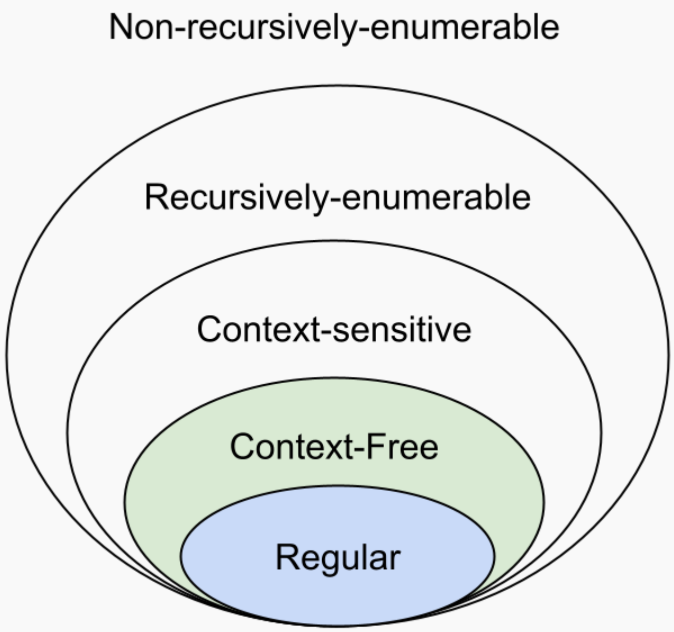
Remember our hierarchy of languages

Chomsky Hierarchy



You've mastered regular ~~expressions~~ languages.

Chomsky Hierarchy



Now what about the next level up?

Context-Free Languages

Example

Regular Language : Finite # of
∪, •, *

- $V = \{S\}$
- $T = \{a, b\}$
- $P = \{S \to \epsilon \mid a \mid b \mid aSa \mid bSb\}$
  (abbrev. for $S \to \epsilon, S \to a, S \to b, S \to aSa, S \to bSb$)

Example

L = {palindromes}

- $V = \{S\}$
- $T = \{a, b\}$
- $P = \{S \to \epsilon \mid a \mid b \mid aSa \mid bSb\}$
  (abbrev. for $S \to \epsilon, S \to a, S \to b, S \to aSa, S \to bSb$)

w_1 w_2

[A B]

aaa

aabaa

abba

$$S \rightsquigarrow aSa \rightsquigarrow abSba \rightsquigarrow abbSbba \rightsquigarrow abb\,b\,bba$$

CF Lang includes any string that can be derived from these production rules

Example

- $V = \{S\}$
- $T = \{a, b\}$
- $P = \{S \to \epsilon \mid a \mid b \mid aSa \mid bSb\}$
  (abbrev. for $S \to \epsilon, S \to a, S \to b, S \to aSa, S \to bSb$)

$$S \rightsquigarrow aSa \rightsquigarrow abSba \rightsquigarrow abbSbba \rightsquigarrow abb\,b\,bba$$

What strings can S generate like this?

Context Free Grammar (CFG) Definition

Definition
A CFG is a quadruple $G = (V, T, P, S)$

- V is a finite set of non-terminal symbols (variables)

$$G = \Big(\quad \text{Variables}, \quad \text{Terminals}, \quad \text{Productions}, \quad \text{Start var} \quad \Big)$$

Context Free Grammar (CFG) Definition

Definition
A CFG is a quadruple $G = (V, T, P, S)$

- V is a finite set of non-terminal symbols
- T is a finite set of terminal symbols (alphabet) Σ

$$G = \left(\text{Variables}, \quad \text{Terminals}, \quad \text{Productions}, \quad \text{Start var} \right)$$

Context Free Grammar (CFG) Definition

Definition
A CFG is a quadruple $G = (V, T, P, S)$

- V is a finite set of non-terminal symbols $= \{A, B, C, S\}$
- T is a finite set of terminal symbols (alphabet) $\{a, b, c, d\}$
- P is a finite set of productions, each of the form

  $A \to \alpha$ → string that is a result of variables & terminals

  $\alpha = aaAbbBC$

  where $A \in V$ and α is a string in $(V \cup T)^*$.

  Formally, $P \subset V \times (V \cup T)^*$.

left side is always a single variable

$G = \big($ Variables, Terminals, Productions, Start var $\big)$

Context Free Grammar (CFG) Definition

Definition
A CFG is a quadruple $G = (V, T, P, S)$

- V is a finite set of non-terminal symbols
- T is a finite set of terminal symbols (alphabet)
- P is a finite set of productions, each of the form
  $A \rightarrow \alpha$
  where $A \in V$ and α is a string in $(V \cup T)^*$.
  Formally, $P \subset V \times (V \cup T)^*$.
- $S \in V$ is a start symbol

$$G = \Big(\text{Variables}, \quad \text{Terminals}, \quad \text{Productions}, \quad \text{Start var} \Big)$$

Example formally...

L = {palindromes}

- $V = \{S\}$
- $T = \{a, b\}$
- $P = \{S \to \epsilon \mid a \mid b \mid aSa \mid bSb\}$
  (abbrev. for $S \to \epsilon, S \to a, S \to b, S \to aSa, S \to bSb$)

$$G = \left(\{S\}, \quad \{a, b\}, \quad \left\{ \begin{array}{c} S \to \epsilon, \\ S \to a, \\ S \to b \\ S \to aSa \\ S \to bSb \end{array} \right\} \quad S \right)$$

Examples

$L = \{0^n 1^n \mid n \geq 0\}$

$V = \{S\}$ $T = \{0, 1\}$ $S = S$

$P = \{ S \to 0S1 \mid \varepsilon \}$

Examples

$L = \{0^n 1^n \mid n \geq 0\}$

$S \to \epsilon \mid 0S1$

Definition
The language generated by CFG $G = (V, T, P, S)$ is denoted by $L(G)$ where $L(G) = \{w \in T^* \mid S \rightsquigarrow^* w\}$.

$L = \{ \varepsilon, 01, 0011, \ldots \}$

Definition
The language generated by CFG $G = (V, T, P, S)$ is denoted by $L(G)$ where $L(G) = \{w \in T^* \mid S \leadsto^* w\}$.

Definition
A language L is context free (CFL) if it is generated by a context free grammar. That is, there is a CFG G such that $L = L(G)$.

Example

$L = \{0^n 1^n \mid n \geq 0\}$

$S \to \epsilon \mid 0S1$

$L = \{0^n 1^m \mid m > n\}$

$V = \{S, A\}$
$T = \{0, 1\}$
$S = S$

$S \to A1$
$A \to 0A1 \mid \varepsilon \mid A1$

00A111

A1

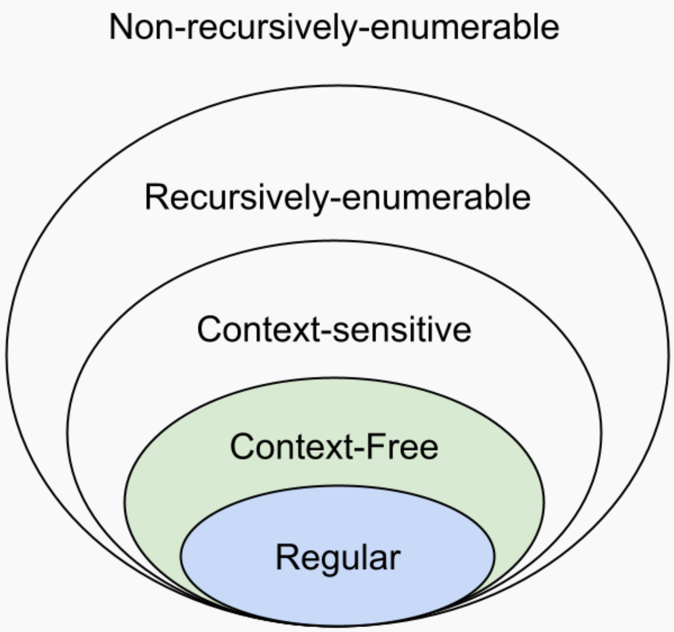
$L = \left\{ w \in \{(,)\}^* \;\middle|\; w \text{ is properly nested string of parenthesis} \right\}.$

$S \to \varepsilon \mid (S)S$

recursive - regexp

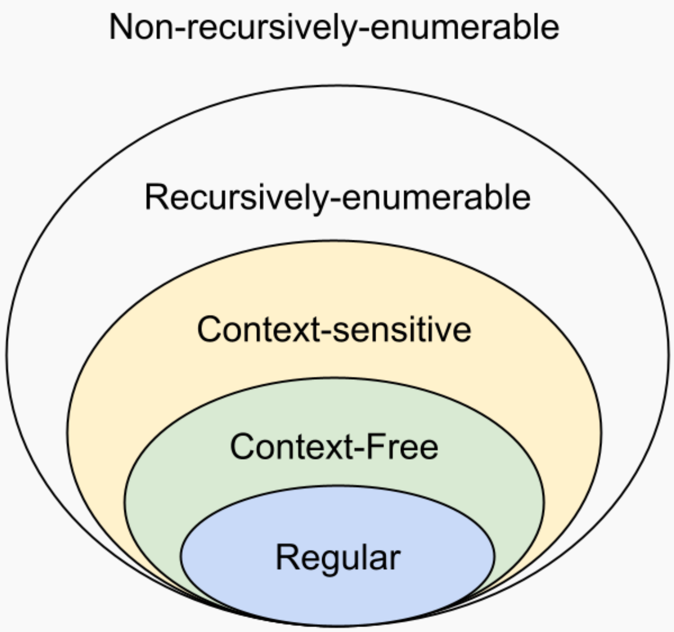
Context-Sensitive Langauges

Chomsky Hierarchy



Now that we ~~mastered~~ acknowledged Context-Free Languages.....

Chomsky Hierarchy



On to the next one.....

Example

The language $L = \{a^n b^n c^n | n \geq 1\}$ is not a context free language.

Prove via pumping lemma

Example

The language $L = \{a^n b^n c^n | n \geq 1\}$ is not a context free language. *but it is a context-sensitive language!*

- $V = \{S, A, B\}$
- $T = \{a, b, c\}$
- $P = \left\{ \begin{array}{c} S \rightarrow abc|aAbc, \\ Ab \rightarrow bA, \\ Ac \rightarrow Bbcc \\ bB \rightarrow Bb \\ aB \rightarrow aa|aaA \end{array} \right\}$

Example

The language $L = \{a^n b^n c^n | n \geq 1\}$ is not a context free language. *but it is a context-sensitive language!*

- $V = \{S, A, B\}$
- $T = \{a, b, c\}$
- $P = \left\{ \begin{array}{c} S \rightarrow abc|aAbc, \\ Ab \rightarrow bA, \\ Ac \rightarrow Bbcc \\ bB \rightarrow Bb \\ aB \rightarrow aa|aaA \end{array} \right\}$

$$S \rightsquigarrow aAbc \rightsquigarrow abAc \rightsquigarrow abBbcc \rightsquigarrow aBbbcc \rightsquigarrow aaAbbcc \rightsquigarrow aabAbcc \rightsquigarrow aabbAcc \rightsquigarrow aabbBbccc \rightsquigarrow aabBbbccc \rightsquigarrow aaBbbbccc \rightsquigarrow aaabbbccc$$

Context ~~Free~~ Sensitive Grammar (C~~F~~SG) Definition

Definition
A CSG is a quadruple $G = (V, T, P, S)$

- V is a finite set of non-terminal symbols
- T is a finite set of terminal symbols (alphabet)
- P is a finite set of productions, each of the form

  $\alpha \rightarrow \beta$

  where β and α are strings in $(V \cup T)^*$.
- $S \in V$ is a start symbol

$$G = \Big(\quad \text{Variables}, \quad \text{Terminals}, \quad \text{Productions}, \quad \text{Start var} \quad \Big)$$

Example formally...

$$L = \{a^n b^n c^n | n \geq 1\}$$

- $V = \{S, A, B\}$
- $T = \{a, b, c\}$
- $P = \left\{ \begin{array}{c} S \rightarrow abc|aAbc, \\ Ab \rightarrow bA, \\ Ac \rightarrow Bbcc \\ bB \rightarrow Bb \\ aB \rightarrow aa|aaA \end{array} \right\}$

$$G = \left(\{S, A, B\}, \quad \{a, b, c\}, \quad \left\{ \begin{array}{c} S \rightarrow abc|aAbc, \\ Ab \rightarrow bA, \\ Ac \rightarrow Bbcc \\ bB \rightarrow Bb \\ aB \rightarrow aa|aaA \end{array} \right\} \quad S \right)$$

Turing Machines

“Most General” computer?

- DFAs are simple model of computation.
- Accept only the regular languages.
- Is there a kind of computer that can accept any language, or compute any function?
- Recall counting argument. Set of all languages: $\{L \mid L \subseteq \{0,1\}^*\}$ is ~~countably infinite~~ / uncountably infinite

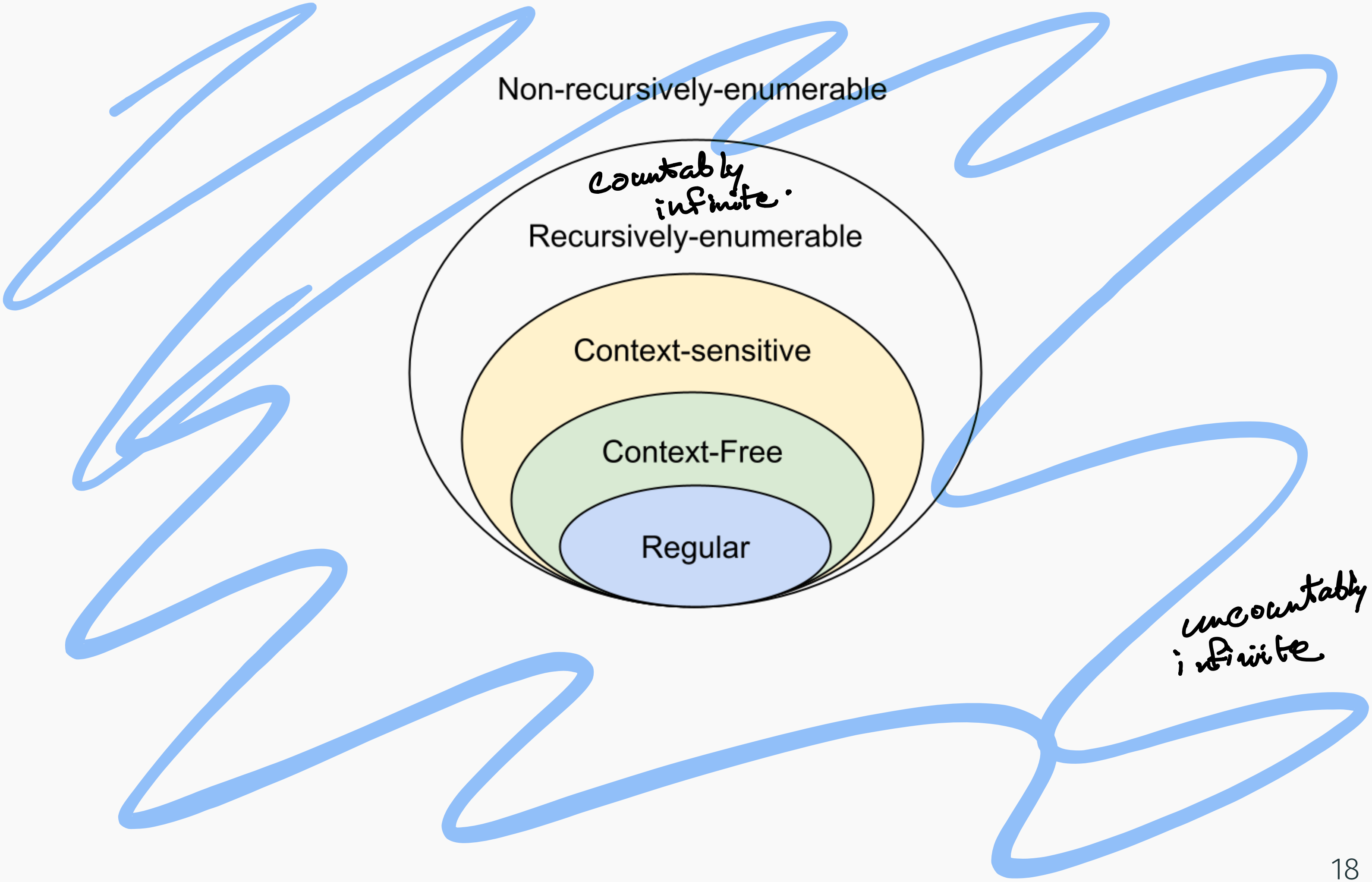
"Most General" computer?

- DFAs are simple model of computation.
- Accept only the regular languages.
- Is there a kind of computer that can accept any language, or compute any function?
- Recall counting argument. Set of all languages: $\{L \mid L \subseteq \{0, 1\}^*\}$ is ~~countably infinite~~ / uncountably infinite
- Set of all programs: $\{P \mid P \text{ is a finite length computer program}\}$: is countably infinite / ~~uncountably infinite.~~

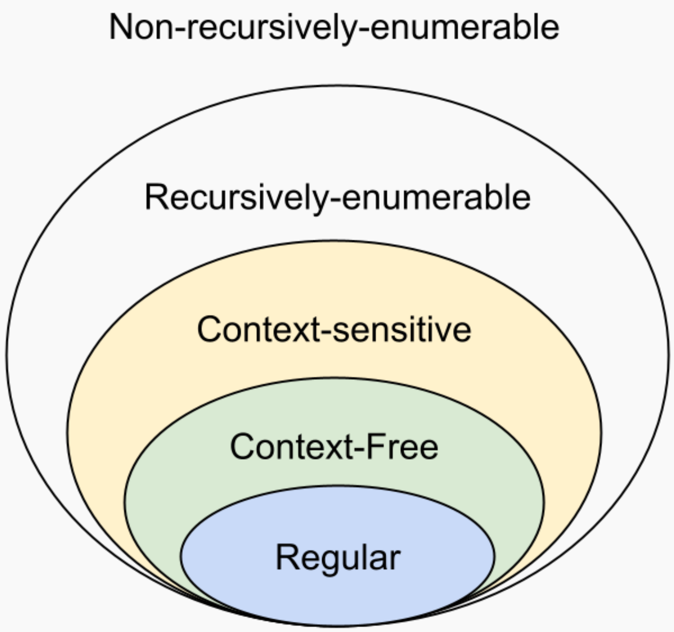
“Most General” computer?

- DFAs are simple model of computation.
- Accept only the regular languages.
- Is there a kind of computer that can accept any language, or compute any function?
- Recall counting argument. Set of all languages: $\{L \mid L \subseteq \{0, 1\}^*\}$ is ~~countably infinite~~ / uncountably infinite
- Set of all programs: $\{P \mid P \text{ is a finite length computer program}\}$: is countably infinite / ~~uncountably infinite~~.
- **Conclusion:** There are languages for which there are no programs.

Chomsky Hierarchy



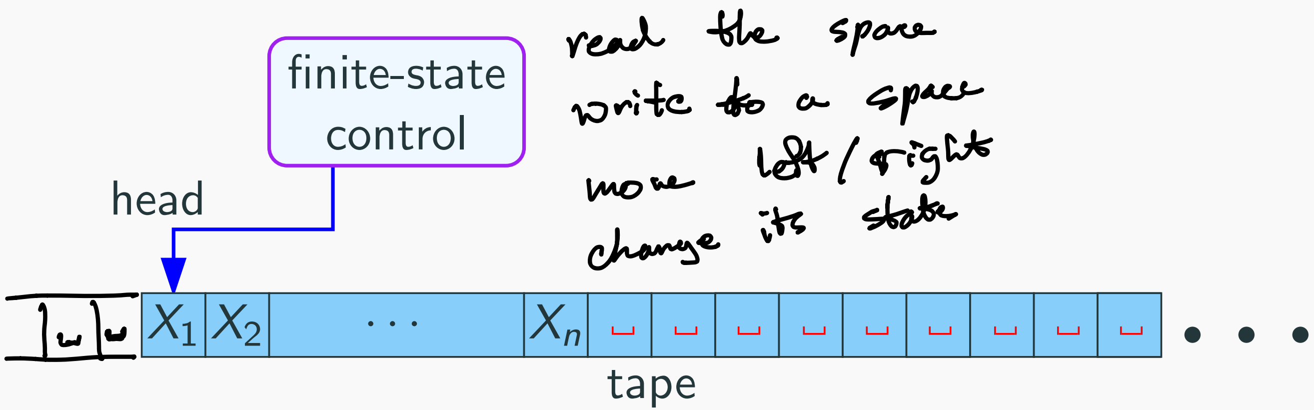
Chomsky Hierarchy



Onto our final class of languages - recursively enumerable (aka Turing-recognizable) languages.

What is a Turing machine

Turing machine



- Input written on (infinite) one sided tape.
- Special blank characters.
- Finite state control (similar to DFA).
- Ever step: Read character under head, write character out, move the head right or left (or stay).

High level goals

- Church-Turing thesis: TMs are the most general computing devices. So far no counter example.
- Every TM can be represented as a string.
- Existence of Universal Turing Machine which is the model/inspiration for stored program computing. UTM can simulate any TM
- Implications for what can be computed and what cannot be computed

next lecture

Turing machine: Formal definition

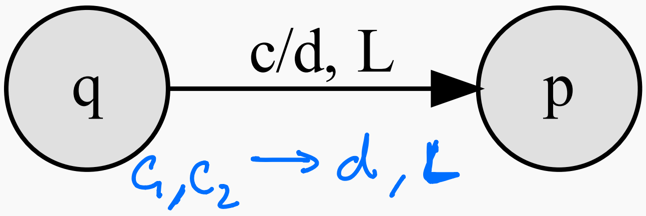
A *Turing machine* is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\mathrm{acc}}, q_{\mathrm{rej}})$

- Q: finite set of states.
- Σ: finite input alphabet.
- Γ: finite tape alphabet. includes "$\sqcup$" = blank space
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\mathsf{L}, \mathsf{R}, \mathsf{S}\}$: Transition function.
- $q_0 \in Q$ is the initial state.
- $q_{\mathrm{acc}} \in Q$ is the *accepting/final* state.
- $q_{\mathrm{rej}} \in Q$ is the *rejecting* state.
- $\sqcup$ or : Special blank symbol on the tape.

Turing machine: Transition function

$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{\mathsf{L}, \mathsf{R}, \mathsf{S}\}$$

As such, the transition



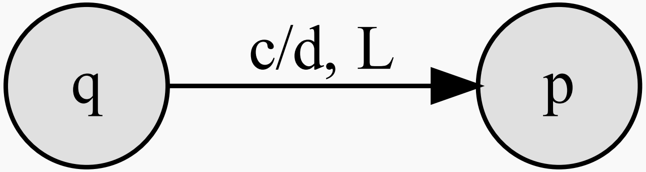
$$\delta(q, c) = (p, d, \mathsf{L})$$

- q: current state.
- c: character under tape head.
- p: new state.
- d: character to write under tape head
- L: Move tape head left.

Turing machine: Transition function

$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{\mathsf{L}, \mathsf{R}, \mathsf{S}\}$$

As such, the transition



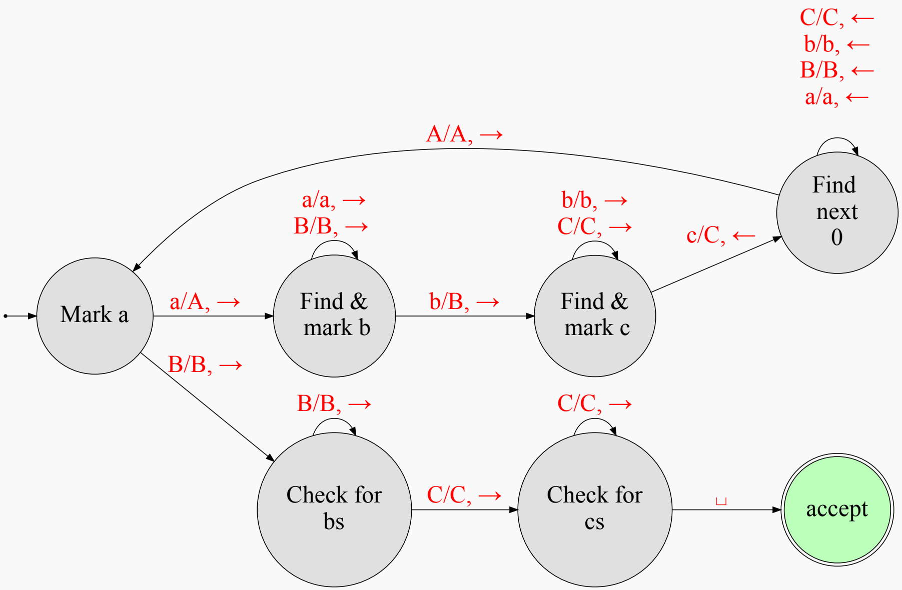
$$\delta(q, c) = (p, d, \mathsf{L})$$

- q: current state.
- c: character under tape head.
- p: new state.
- d: character to write under tape head
- L: Move tape head left.

Missing transitions lead to hell state. "Blue screen of death." "Machine crashes."

Some examples of Turing machines

Example: Turing machine for $a^n b^n c^n$



Can view this Turing machine in action on `turingmachine.io`!

Languages defined by a Turing machine

Recursive vs. Recursively Enumerable

- *Recursively enumerable* (aka *RE*) languages

$$L = \{L(M) \mid M \text{ some Turing machine}\}.$$

- *Recursive* / *decidable* languages

$$L = \{L(M) \mid M \text{ some Turing machine that halts on all inputs}\}.$$

Recursive vs. Recursively Enumerable

- *Recursively enumerable* (aka *RE*) languages (bad)

$$L = \{L(M) \mid M \text{ some Turing machine}\}.$$

- *Recursive* / *decidable* languages (good)

$$L = \{L(M) \mid M \text{ some Turing machine that halts on all inputs}\}.$$

Recursive vs. Recursively Enumerable

- *Recursively enumerable* (aka *RE*) languages (bad)

$$L = \{L(M) \mid M \text{ some Turing machine}\}.$$

- *Recursive / decidable* languages (good)

$$L = \{L(M) \mid M \text{ some Turing machine that halts on all inputs}\}.$$

- Fundamental questions:
  - What languages are RE?
  - Which are recursive?
  - What is the difference?
  - What makes a language decidable?