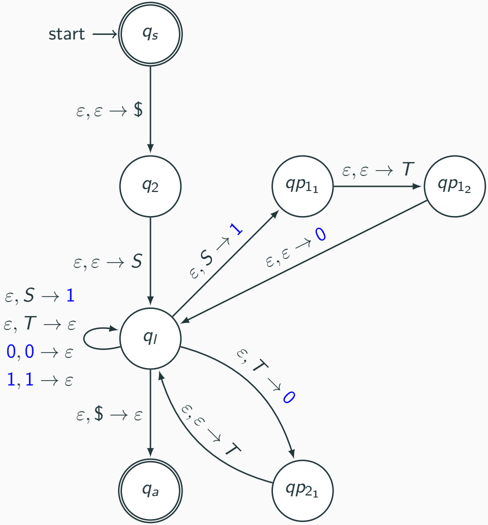
Pre-lecture brain teaser

What is the context-free grammar of the following push-down automata:



ECE-374-B: Lecture 8 - Context-sensitive and decidable languages

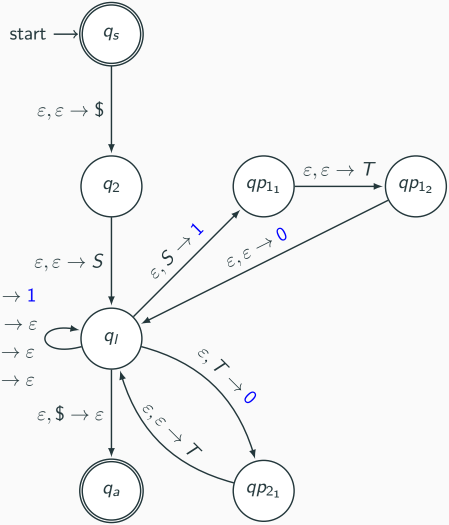
Instructor: Nickvash Kani

Febuary 10, 2022

University of Illinois at Urbana-Champaign

Pre-lecture brain teaser

What is the context-free grammar of the following push-down automata:



Closure properties of CFLs

Closure Properties of CFLs

$G_1 = (V_1, T, P_1, S_1)$ and $G_2 = (V_2, T, P_2, S_2)$
Assumption: $V_1 \cap V_2 = \emptyset$, that is, non-terminals are not shared

Closure Properties of CFLs

$G_1 = (V_1, T, P_1, S_1)$ and $G_2 = (V_2, T, P_2, S_2)$
Assumption: $V_1 \cap V_2 = \emptyset$, that is, non-terminals are not shared

Theorem

CFLs are closed under union. L_1, L_2 CFLs implies $L_1 \cup L_2$ is a CFL.

Theorem

CFLs are closed under concatenation. L_1, L_2 CFLs implies $L_1 \bullet L_2$ is a CFL.

Theorem

CFLs are closed under Kleene star.

If L is a CFL $\implies$ L^* is a CFL.

Closure Properties of CFLs- Union

$G_1 = (V_1, T, P_1, S_1)$ and $G_2 = (V_2, T, P_2, S_2)$
Assumption: $V_1 \cap V_2 = \emptyset$, that is, non-terminals are not shared.

Theorem

CFLs are closed under union. L_1, L_2 *CFLs implies* $L_1 \cup L_2$ *is a CFL.*

Closure Properties of CFLs- Concatenation

Theorem

CFLs are closed under concatenation. L_1, L_2 *CFLs implies* $L_1 \bullet L_2$ *is a CFL.*

Closure Properties of CFLs- Kleene star

Theorem

CFLs are closed under Kleene star.

If L is a CFL $\implies L^*$ is a CFL.

Bad news: Canonical non-CFL

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

Proof based on pumping lemma for CFLs. See supplemental for the proof.

More bad news: CFL not closed under intersection

Theorem

CFLs are not closed under intersection.

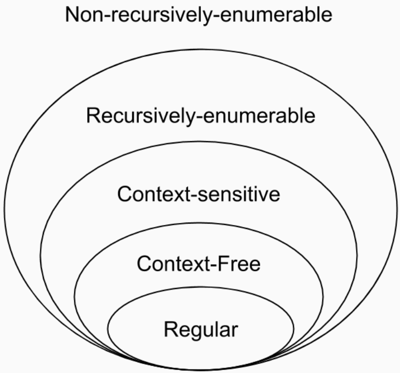
Even more bad news: CFL not closed under complement

Theorem

CFLs are not closed under complement.

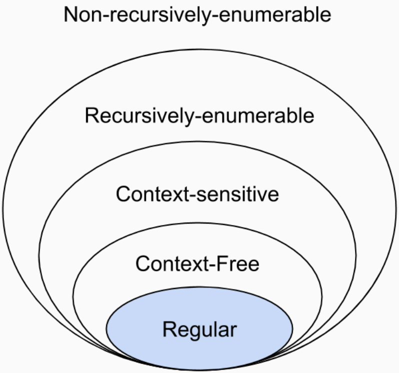
Larger world of languages!

Chomsky Hierarchy



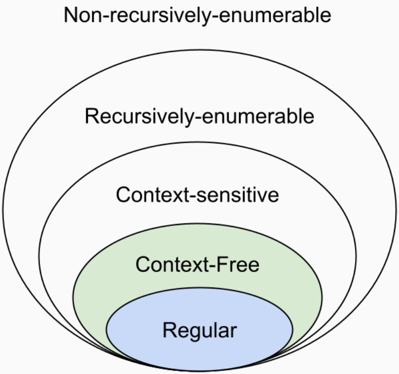
Remember our hierarchy of languages

Chomsky Hierarchy

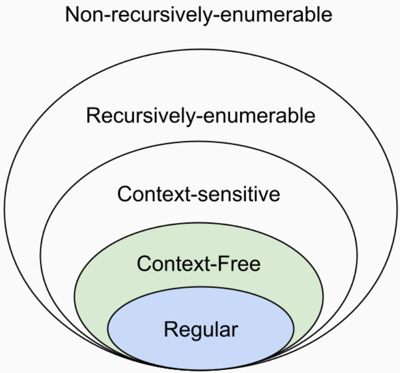


You've mastered regular expressions.

Chomsky Hierarchy

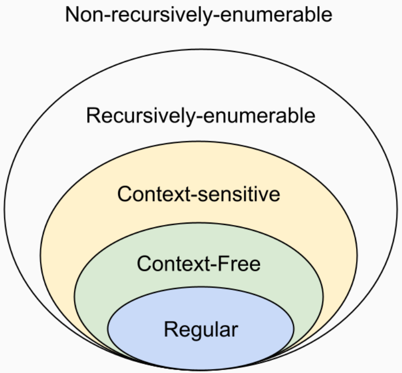


Chomsky Hierarchy



Now what about the next level up?

Chomsky Hierarchy



On to the next one.....

Context-Sensitive Languages

Example

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

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!

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

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!

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

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

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

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

Example formally...

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

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

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

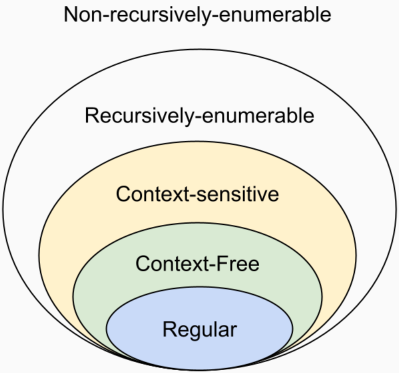
Other examples of context-sensitive languages

$$L_{Cross} = \{a^m b^n c^m d^n | m, n \geq 1\} \tag{1}$$

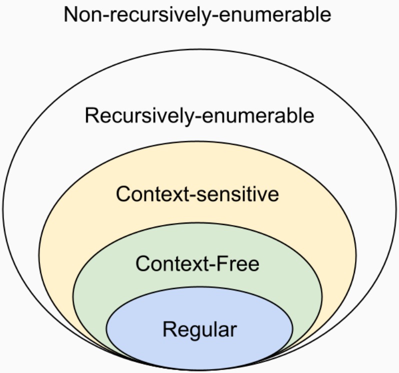
Turing Machines

What is a Turing machine

Chomsky Hierarchy

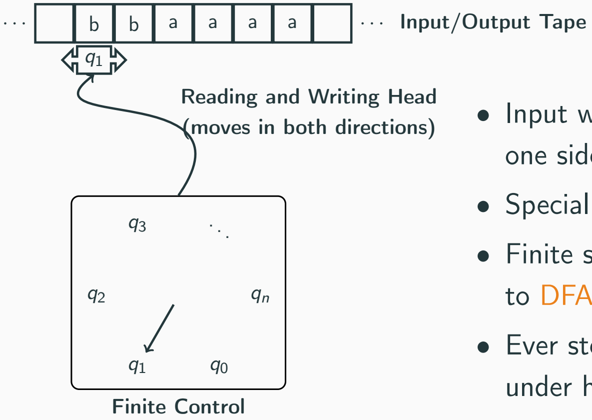


Chomsky Hierarchy



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

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

- TMs are the most general computing devices.
- Church-Turing thesis: All sufficiently complicated machines are <u>equivalent</u> to Turing machines. This includes (but is not limited to): Lambda Calculus, RAM machines, etc.
- Strong Church-Turing thesis: the transformations between these are <u>efficient</u> (polynomial time overhead)
- 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

Examples of Turing machines

- binary increment

Turing machine: Formal definition

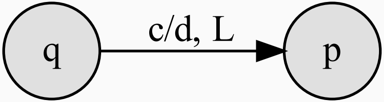
A Turing machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\mathrm{acc}}, q_{\mathrm{rej}})$

- Q: finite set of states.
- Σ: finite input alphabet.
- Γ: finite tape alphabet.
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\mathtt{L}, \mathtt{R}, \mathtt{S}\}$: 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 $\textvisiblespace$: Special blank symbol on the tape.

Turing machine: Transition function

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

As such, the transition



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

- q: current state.
- c: character under tape head.
- p: new state.
- d: character to write under tape head
- $\mathtt{L}$: Move tape head left.

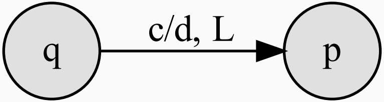
Can also be written as

$$c \to d, L \qquad (2)$$

Turing machine: Transition function

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

As such, the transition



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

- q: current state.
- c: character under tape head.
- p: new state.
- d: character to write under tape head
- $\mathtt{L}$: Move tape head left.

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

Some examples of Turing machines

turingmachine.io

- equal strings TM
- palindrome TM

Languages defined by a Turing machine

Language defined by a turing machine

- Language accepted by a Turing machine

$$L(M) = \{x \in \Sigma \mid \text{ on input } x, M \text{ reaches } q_{acc} \text{ and halts}\}.$$

- If $x \notin L(M)$,
  - M might reject M by reaching q_{rej}
  - or M might not halt at all, M diverges on x.

Recursive vs. Recursively Enumerable

- Recursively enumerable (aka RE, aka semi-decidable) languages

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

- Recursive / decidable languages

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

Recursive vs. Recursively Enumerable

- Recursively enumerable (aka RE, aka semi-decidable) languages (bad)

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

- Recursive / decidable languages (good)

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

Recursive vs. Recursively Enumerable

- Recursively enumerable (aka RE, aka semi-decidable) languages (bad)

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

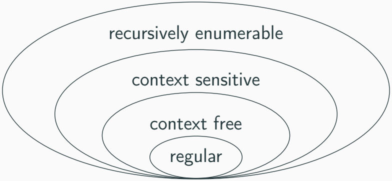
- Recursive / decidable languages (good)

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

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

Well that was a journey....

Zooming out



Grammar	Languages	Production Rules	Automation	Examples
Type-0	Turing machine	$\gamma \to \alpha$ (no constraints)	Turing machine	$L = \{w \mid M$ is a TM that halts on $w\}$
Type-1	Context-sensitive	$\alpha A \beta \to \alpha \gamma \beta$	Linear bounded Non-deterministic Turing machine	$L = \{a^n b^n c^n \mid n > 0\}$
Type-2	Context-free	$A \to \alpha$	Non-deterministic Push-down automata	$L = \{a^n b^n \mid n > 0\}$
Type-3	Regular	$A \to aB$	Finite State Machine	$L = \{a^n \mid n > 0\}$

1

Meaning of symbols:

- a = terminal
- A, B = variables
- α, β, γ = string of $\{a \cup A\}^*$
- α, β = maybe empty —— γ = never empty