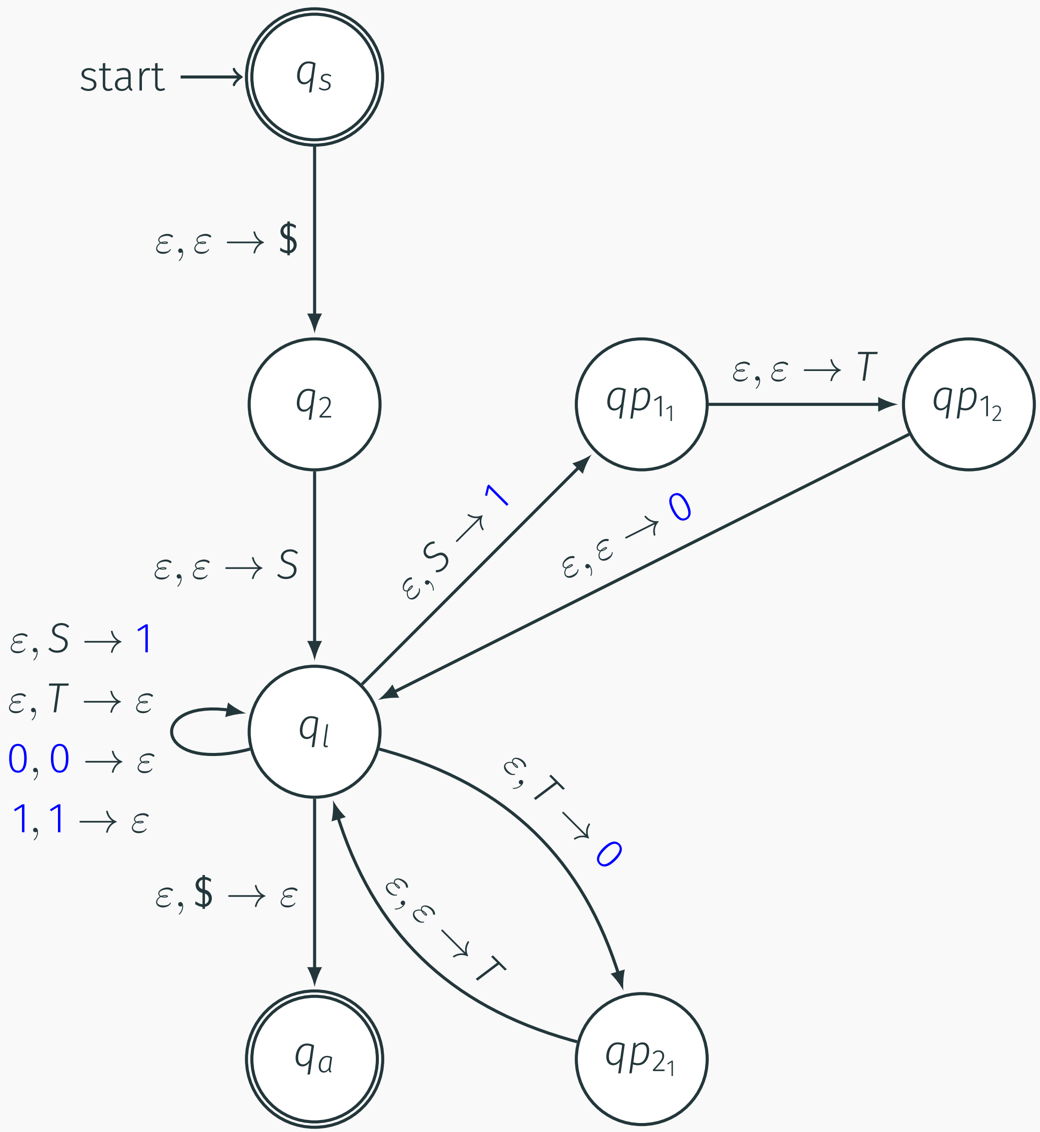
Pre-lecture brain teaser

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

February 9, 2023

University of Illinois at Urbana-Champaign

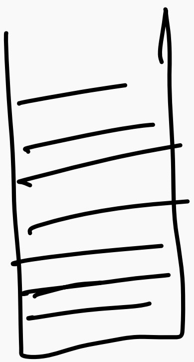
Pre-lecture brain teaser

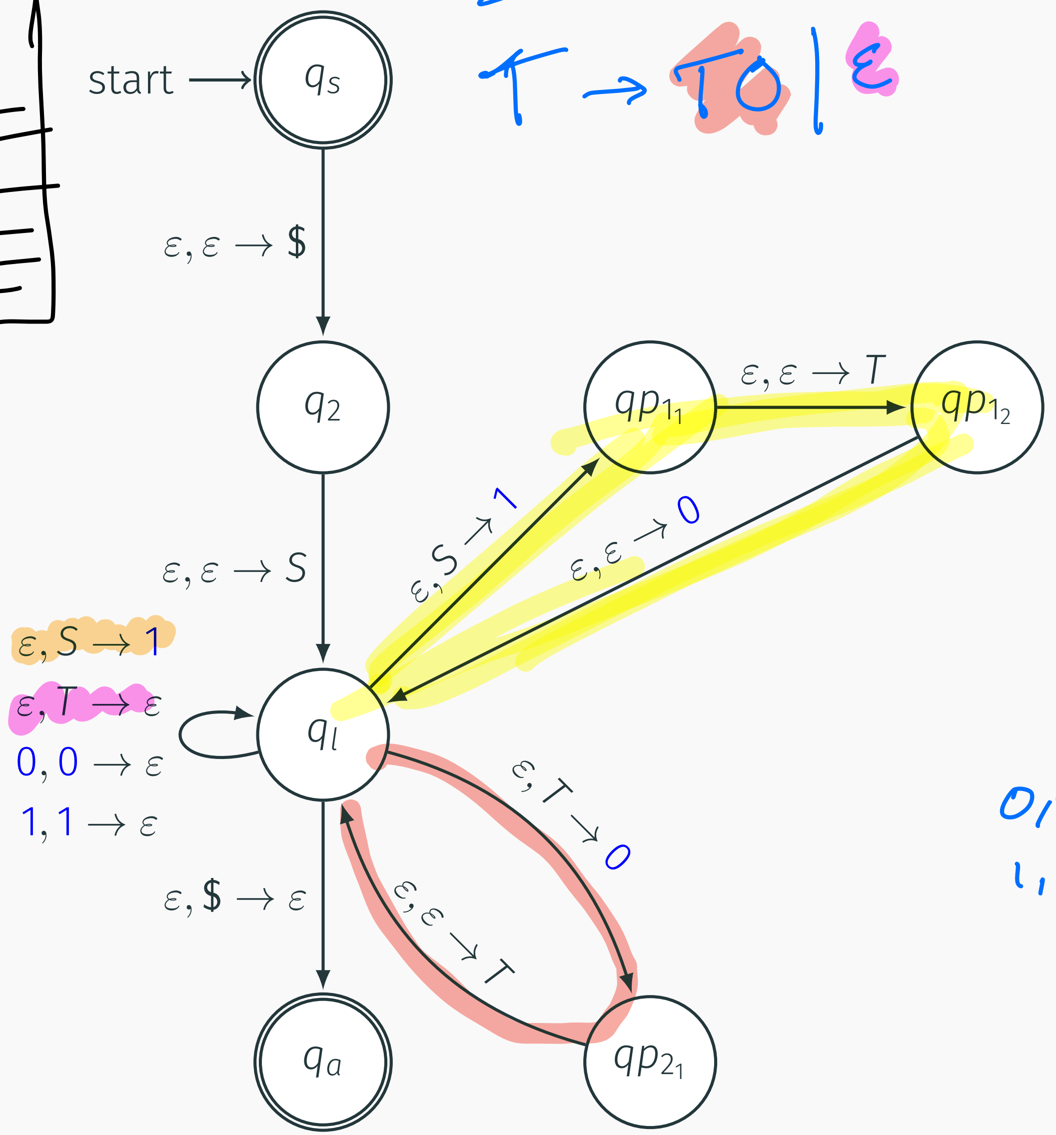
What is the context-free grammar of the following push-down automata:

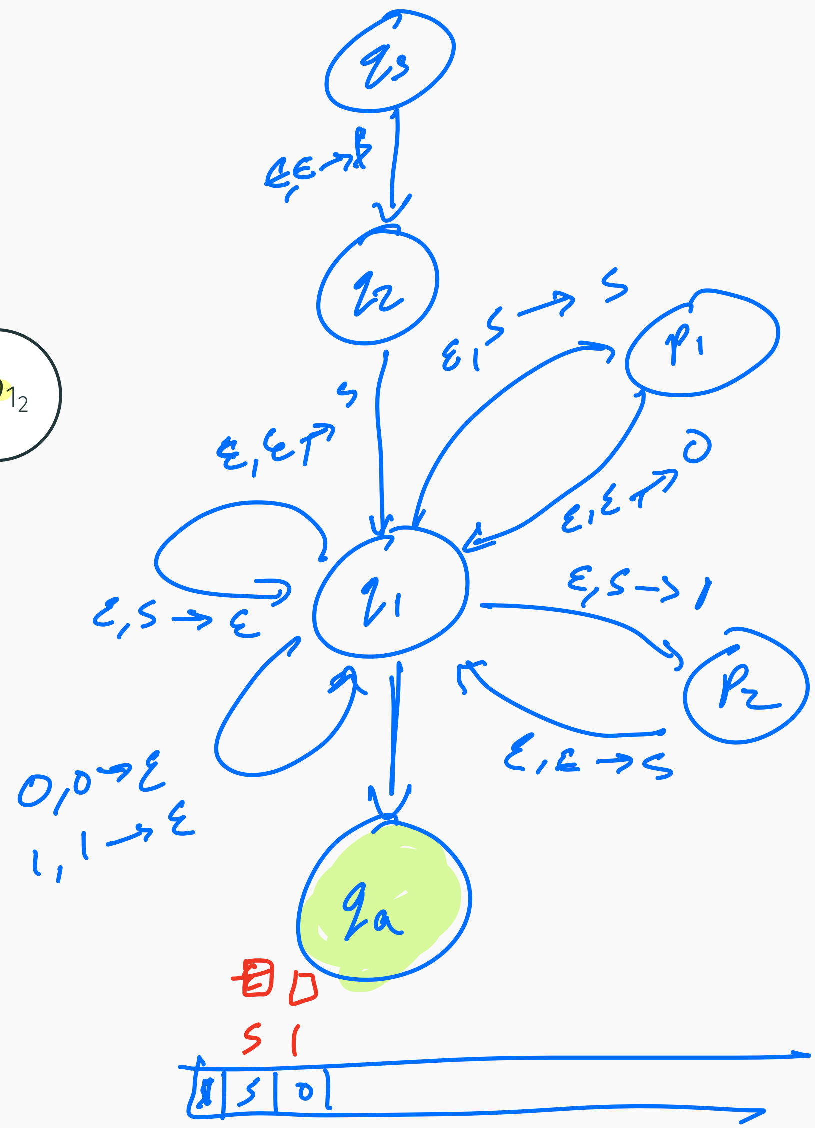
$S \to 0T1 \mid 1$

$T \to T0 \mid \varepsilon$

$S \to 0S \mid S1 \mid \varepsilon$







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

$G_1 = (V_1, T, P_1, S_1)$ 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

$L_3 = L_1 \cup L_2$ $\quad P_3 = \{ S_3 \to S_1 \mid S_2, \ \vdots \}$

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

$L_4 = L_1 \cdot L_2$ $\quad P_4 = \{ S_4 \to S_1 \cdot S_2, \ \vdots \}$

Theorem

CFLs are closed under Kleene star.

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

$L_5 = L_1^*$ $\quad P_5 = \{ S_5 \to S_1 S_5 \mid \varepsilon, \ \vdots \}$

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.

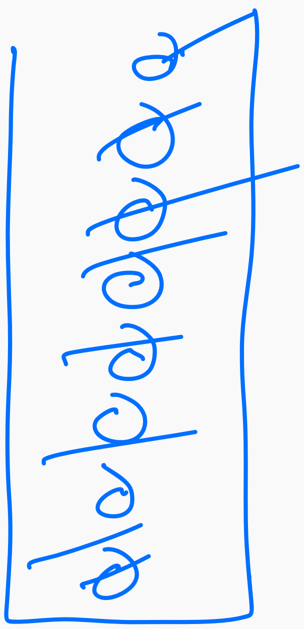
If L is a CFL $\implies$ L^* is a CFL.

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.

$0^n 1^n$ as CFL



More bad news: CFL not closed under intersection

Theorem

CFLs are not closed under intersection.

Are CF

$$L_1 = \{a^n b^n c^m \mid n, m \geq 0\}$$

$$L_2 = \{a^n b^m c^n \mid n, m \geq 0\}$$

$$L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 0\} \leftarrow \text{Not CF}$$

Even more bad news: CFL not closed under complement

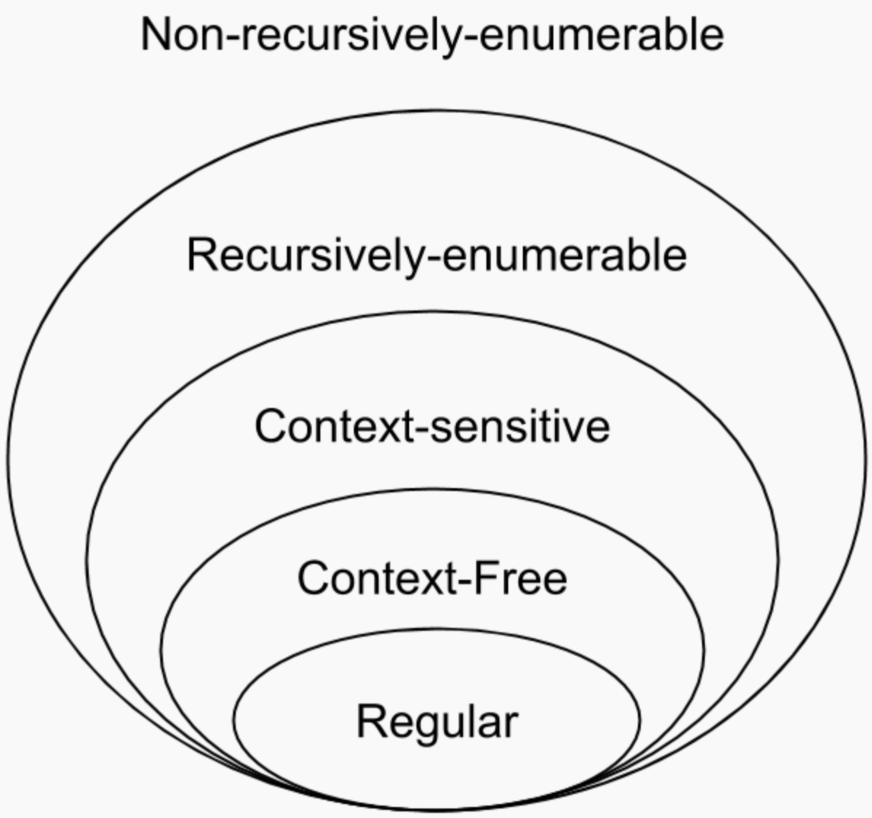
Theorem

CFLs are not closed under complement.

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$$

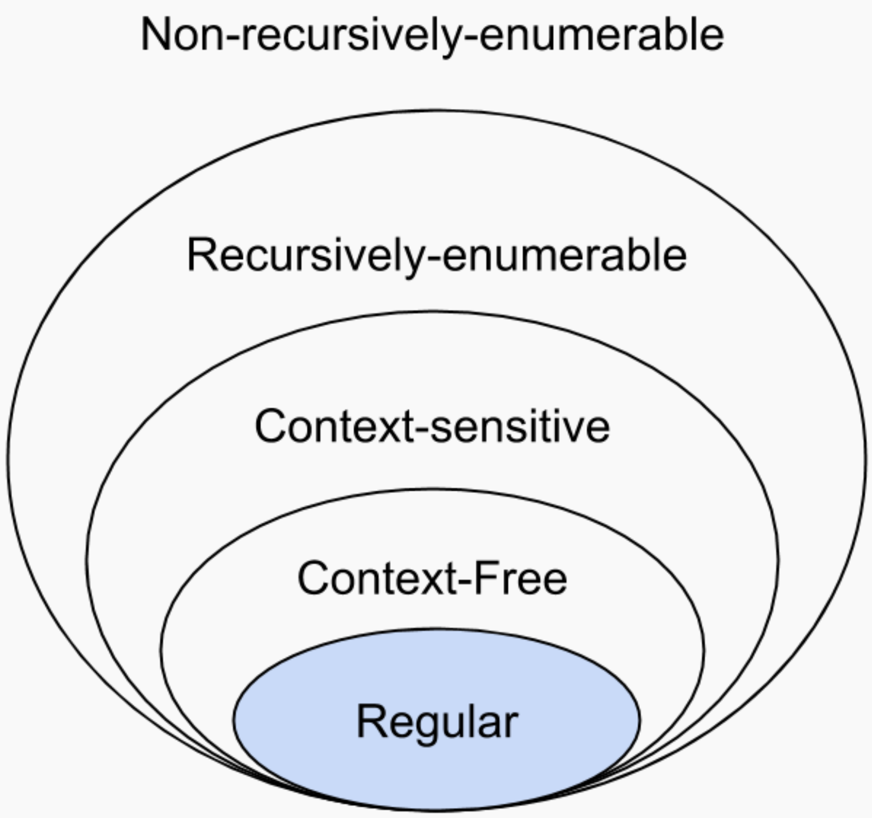
Larger world of languages!

Chomsky Hierarchy



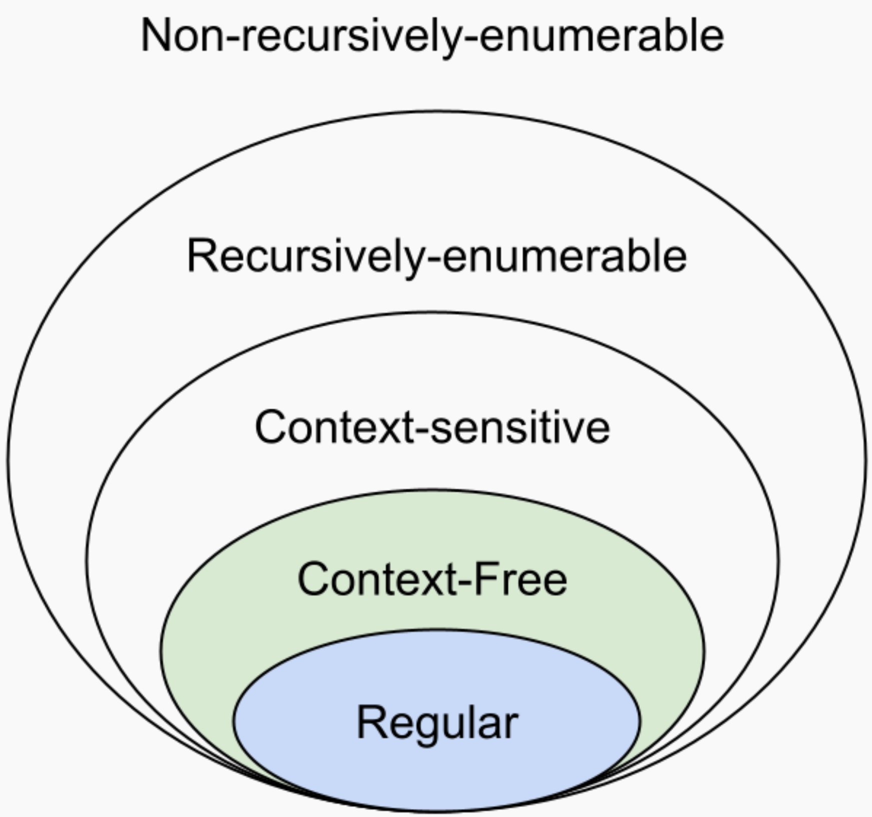
Remember our hierarchy of languages

Chomsky Hierarchy

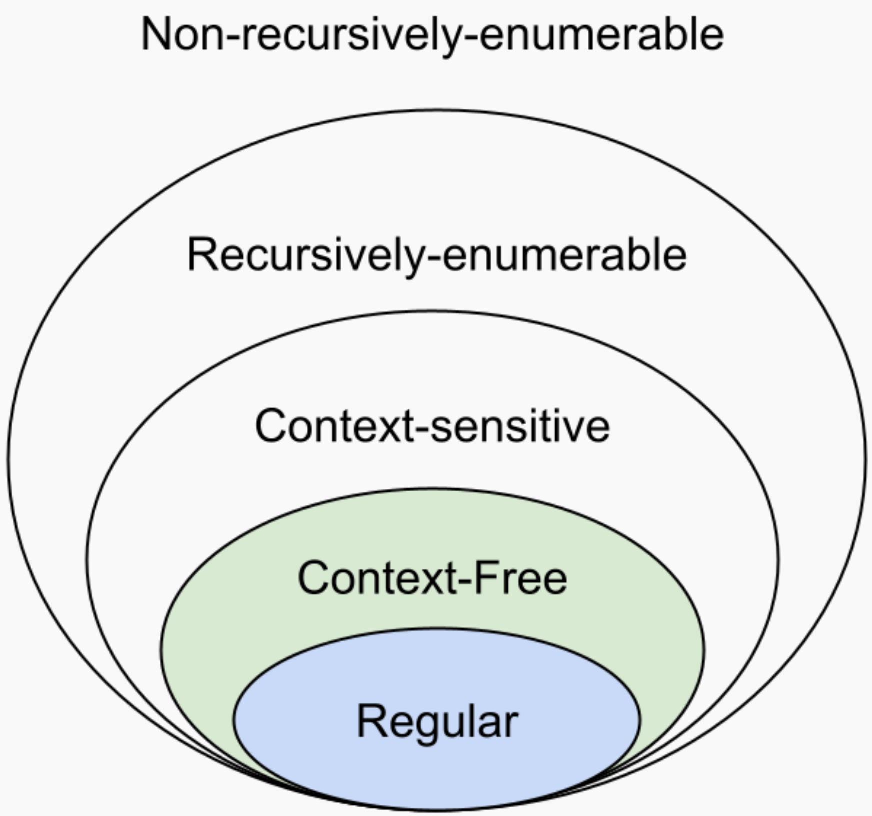


You’ve mastered regular expressions.

Chomsky Hierarchy

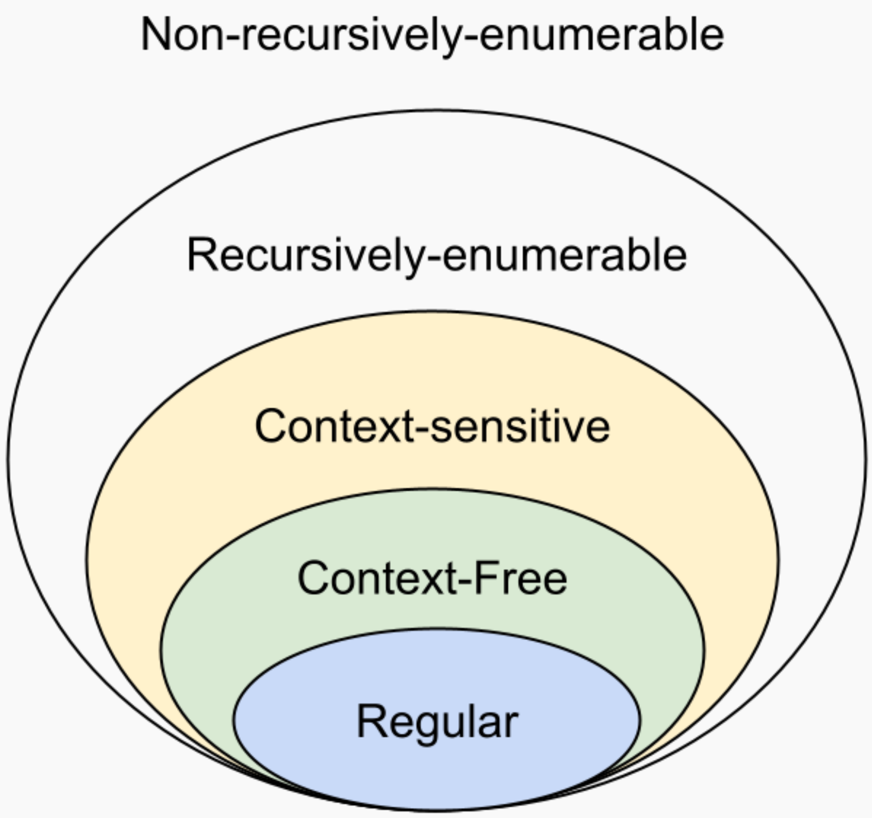


Chomsky Hierarchy



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!*

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

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!*

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

CFG:
$V \rightarrow \{T \cup V\}^*$

CSG:
$\{T \cup V\}^* \rightarrow \{T \cup V\}^*$

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

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

  $\alpha \to \beta$

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

Regular
$V \to T^* V$
Context Free
$V \to \{V+T\}^*$
Context Sensitive
$\{V+T\}^* \to \{V+T\}^*$
Turing Recognizing
non empty

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

Example formally...

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

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

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

Other examples of context-sensitive languages

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

$L_1 = a^m c^m \qquad L_2 = B^n d^n$

$P_1 = \quad S_1 \rightarrow a S_1 C \mid \varepsilon \qquad P_2 : S_2 \rightarrow B S_2 d \mid \varepsilon$

$CB \rightarrow BC$

$B \rightarrow b$

$C \rightarrow c$

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

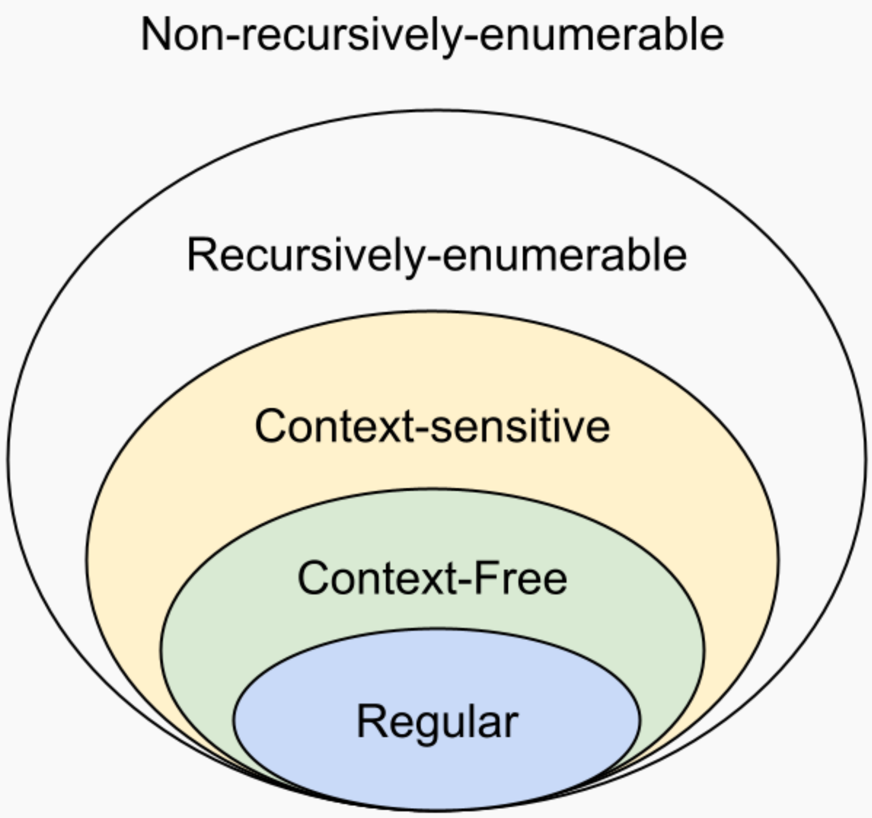
“Most General” computer?

- 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: $\{P \mid P$ is a finite length computer program$\}$: is countably infinite / ~~uncountably infinite.~~

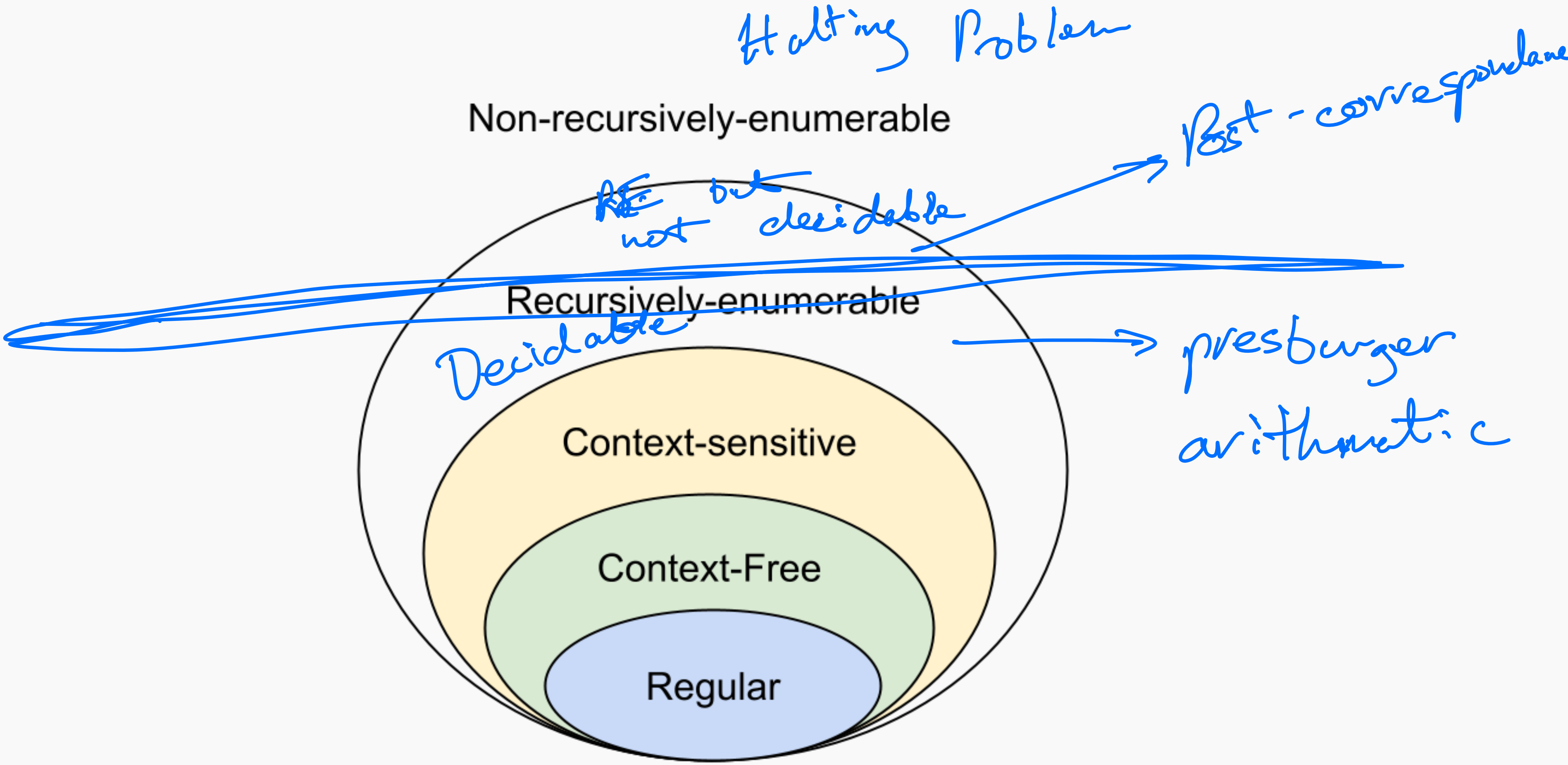
“Most General” computer?

- 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



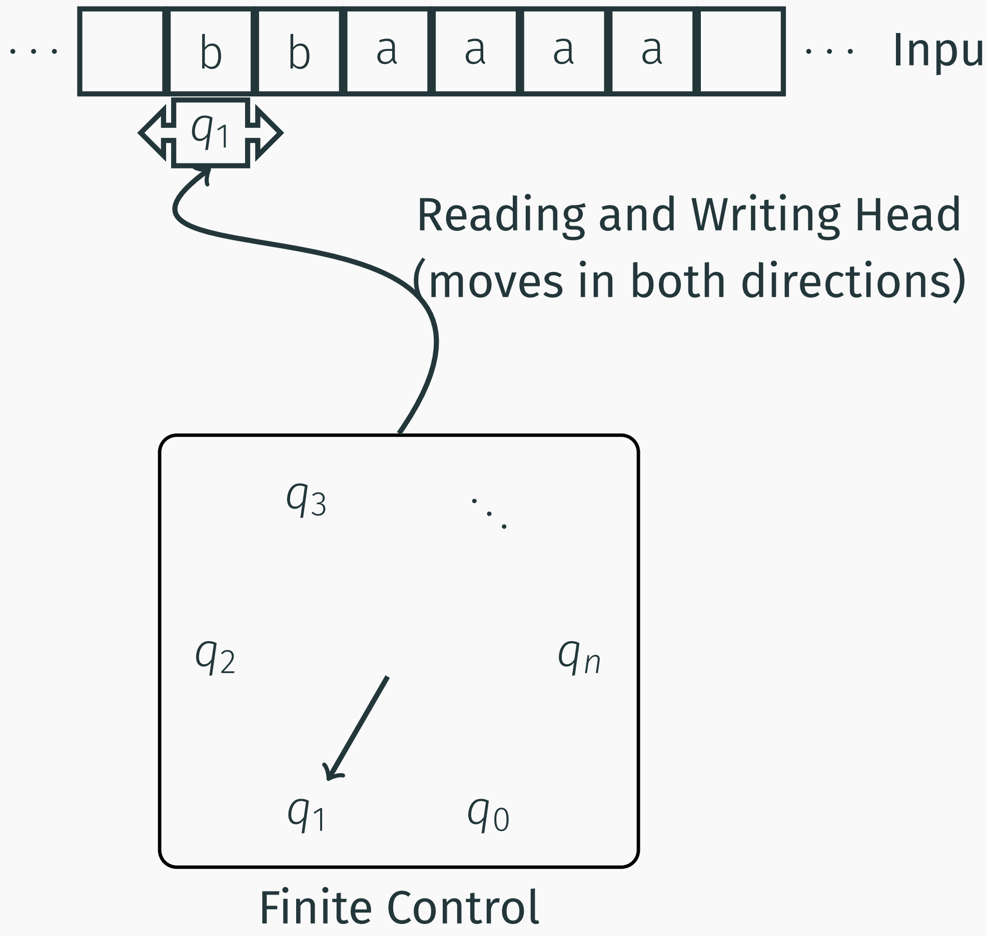
Chomsky Hierarchy



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).
- Every 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 Decidability

Examples of Turing

turingmachine.io

- binary increment

Turing machine: Formal definition

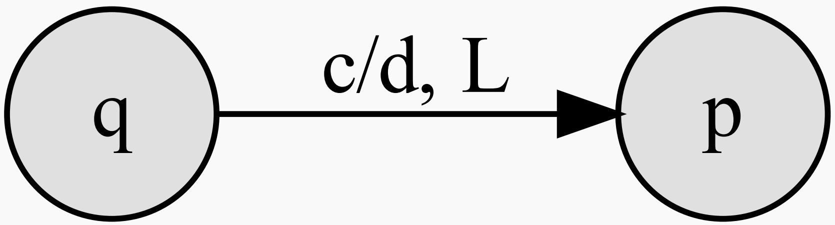
A Turing machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\mathrm{acc}}, q_{\mathrm{rej}})$

- Q: finite set of states.
- Σ: finite input alphabet.
- Γ: finite tape alphabet.
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\mathsf{L}, \mathsf{R}, \mathsf{S}\}$: 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 $\boxed{?}$: 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.

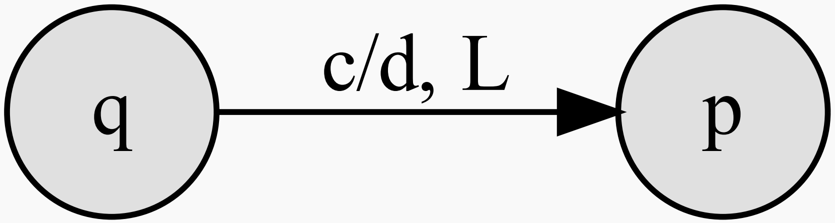
Can also be written as

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

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

turingmachine.io

- equal strings TM
- palindrome TM

Languages defined by a Turing machine

Recursive vs. Recursively Enumerable

- Recursively enumerable (aka RE) languages

  Turing recognizable

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

- 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?

What is Decidable?

Decidable vs recursively-enumerable

A semi-decidable problem (equivalent of recursively enumerable) could be:

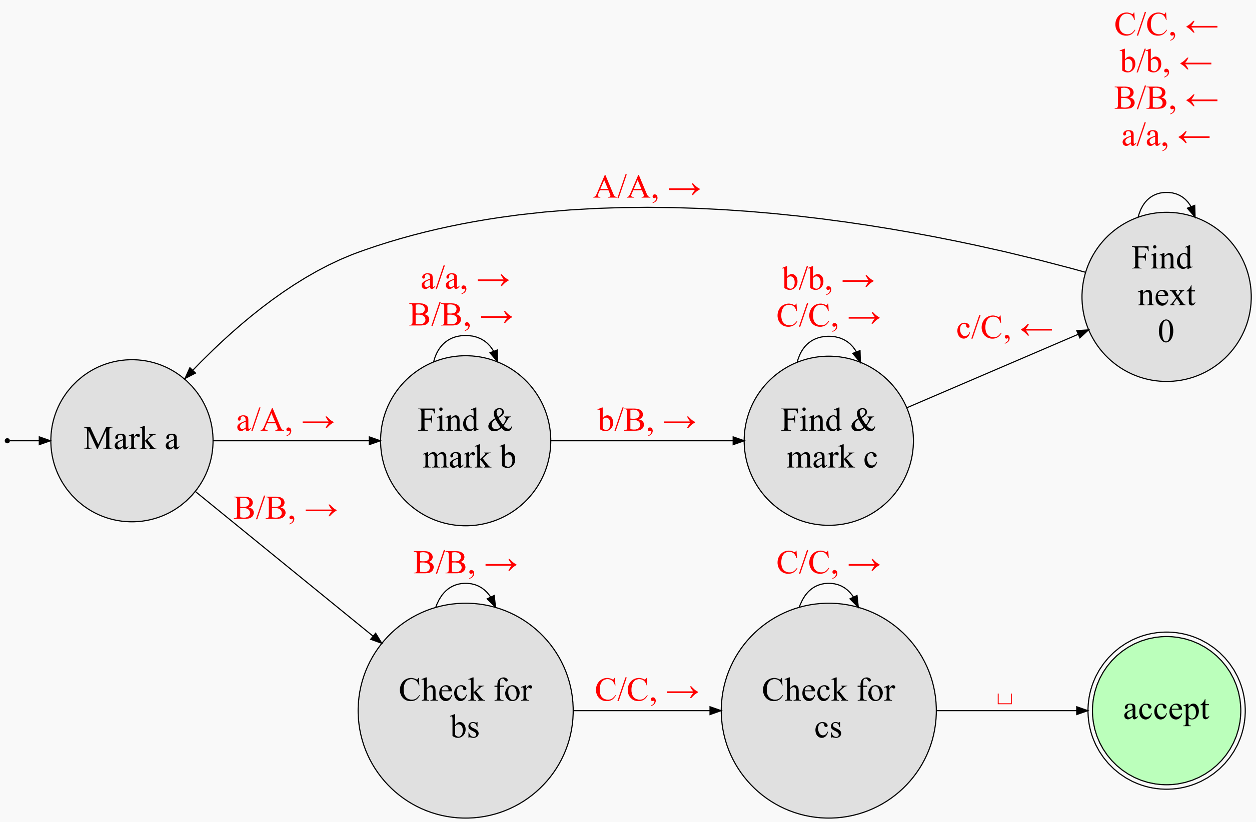
- **Decidable** - equivalent of recursive (TM always accepts or rejects).
- **Undecidable** - Problem is not recursive (doesn't always halt on negative)

There are undecidable problem that are not semi-decidable (recursively enumerable).

Infinite Tapes? Do we need them?

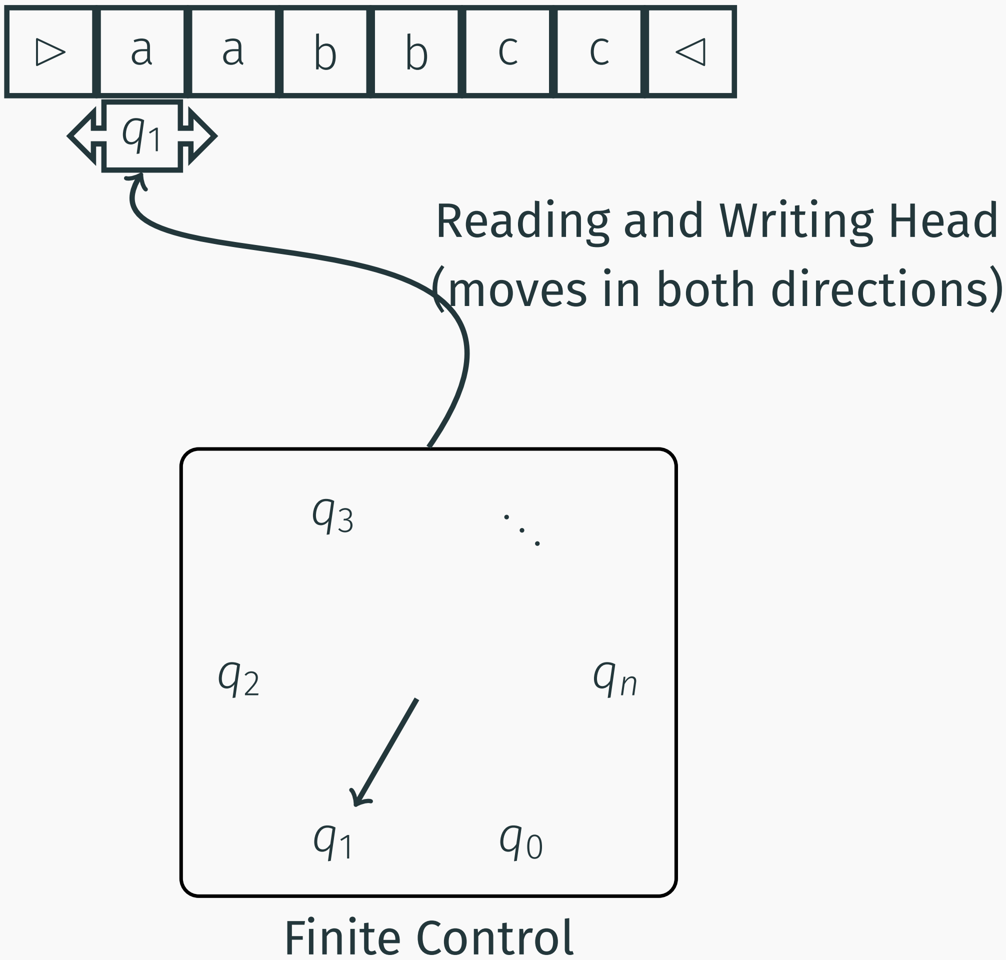
$a^n b^n c^n$

Let's look at the TM that recognizes $L = \{a^n b^n c^n | n \geq 0\}$:



Tape size $\propto$ input size

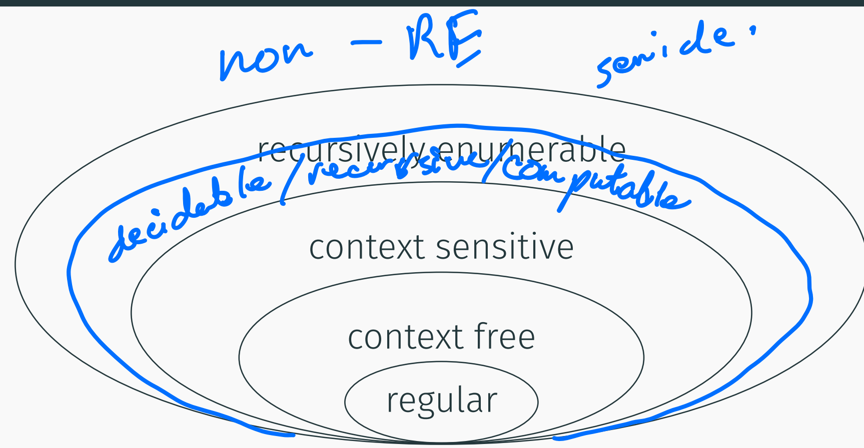
Linear Bounded Automata



- (Nondeterministic) Linear bounded automata can recognize all context sensitive languages.
- Machine can non-deterministically apply all production rule to input in reverse and see if we end up with the start token.

Well that was a journey....

Zooming out



Chomsky hierarchy

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

Meaning of symbols:

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