Turing Machines

Introduction

Turing machines:

- The simplest model of general-purpose computation
- Mathematically define what we **can** compute, Prove we **cannot** compute

Outline

- The Turing Machine
  - Operation
  - Examples
  - Languages
- Variants of Turing Machines
  - Multi-Tape Turing Machines
  - Nondeterministic Turing Machines
  - RAM Machine
  - Two-Stack Pushdown Automata
  - Chomsky Hierarchy

Turing Machine

Finite automata	Turing machine
Read-only input tape	Read-Write (memory) tape
Left-to-right scan of input	Read head can move left and right
Accept/Reject at end of string	Special states to immediately accept/reject

Turing Machine: A Turing machine is a 7-tuple $T = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$, where

- Q is the finite set of states
- Σ is the input alphabet not containing the **blank symbol** $\sqcup \notin \Sigma$
- Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subset \Gamma$
- δ: $Q \times \Gamma \mapsto Q \times \Gamma \times \{L,R\}$ is the transition function
- $q_0 \in Q$ is the start state
- $q_{acc} \in Q$ is the accept state
- $q_{rej} \in Q$ is the reject state, where $q_{rej} \neq q_{acc}$

Turing Machine Execution

LINK[1]: EXAMPLE OF TURING MACHINE CONSTRUCTION

LINK[2]: EXAMPLE OF TURING MACHINE CONSTRUCTION

Note: Deterministic means every state has an edge for every single symbol in Σ

Turing Machine Configurations

The configuration of a Turing machine si given be $\alpha q \beta$, where

- $q \in Q$ is the current control state
- $\alpha \in \Gamma^*$ is the tape contents up to the location before current head position
- $\beta \in \Gamma^*$ is the tape contents from the current head position to the last non-blank character (the head is on the first element of β)

Under the assumption that Q and Γ are disjoint: $Q \cap \Gamma = \emptyset$

Special Configurations

Start Configuration

$C^{[0]} = q_0\sigma$,

where $q_0 \in Q$ is the start state and $\sigma \in \Sigma^*$ is the input string

Accepting Configuration

$C_{acc} = \{\alpha q_{acc} \beta \mid \alpha \in \Gamma^* \wedge \beta \in \Gamma^*\}$,

where $q_{acc} \in Q$ is the accept state

Rejecting Configuration

$C_{rej} = \{\alpha q_{rej} \beta \mid \alpha \in \Gamma^* \wedge \beta \in \Gamma^*\}$,

where $q_{rej} \in Q$ is the reject state

Execution

TM Execution: given string ω, TM T tracks configurations $C_{[0]}$, $C_{[1]}$,... , $C_{[n]}$, where

1. $C^{[0]}$ is the start configuration
2. Each transition goes from $C_{[i]}$ to $C^{[i+1]}$
3. $C^{[n]} \in C_{acc} \cup C_{rej}$) is an accepting or rejecting configuration

TM Behaviours:

1. Enter accept state
2. Enter a reject state
3. Loop forever

Turing Machine Languages

The language **recognized** by Turing machine T is the set of strings which T will enter an accepting configuration.

The language **rejected** by Turing machine T is the set of strings on which T will enter a rejecting configuration.

Recognizer and Decider

Recognizer: Turing machine T is a recognizer for language L if it enters an accept state for every string in L and does not enter an accept state for every string not in L

$L_{\text{acc}}(\text{T}) = \text{L}$

Decider: Turing machine T is a decider for language L if it always enter an accept or reject state, i.e., never loops forever. T decides its language. i

T decides L if $L_{\text{acc}}(\text{T}) = \text{L}$ and $L_{\text{rej}} = \bar{\text{L}}$

For each $\alpha \in \text{L}$ and $\beta \notin \text{L}$:

Recognizer for L	Decider for L
Accepts α	Accepts α
Does not accept β	Rejects β
Can loop forever	Always halts

Turing-Recognizable and Turing-Decidable Languages

Language L is **Turing-recognizable** if there exists some Turing machine T that recognizes L.

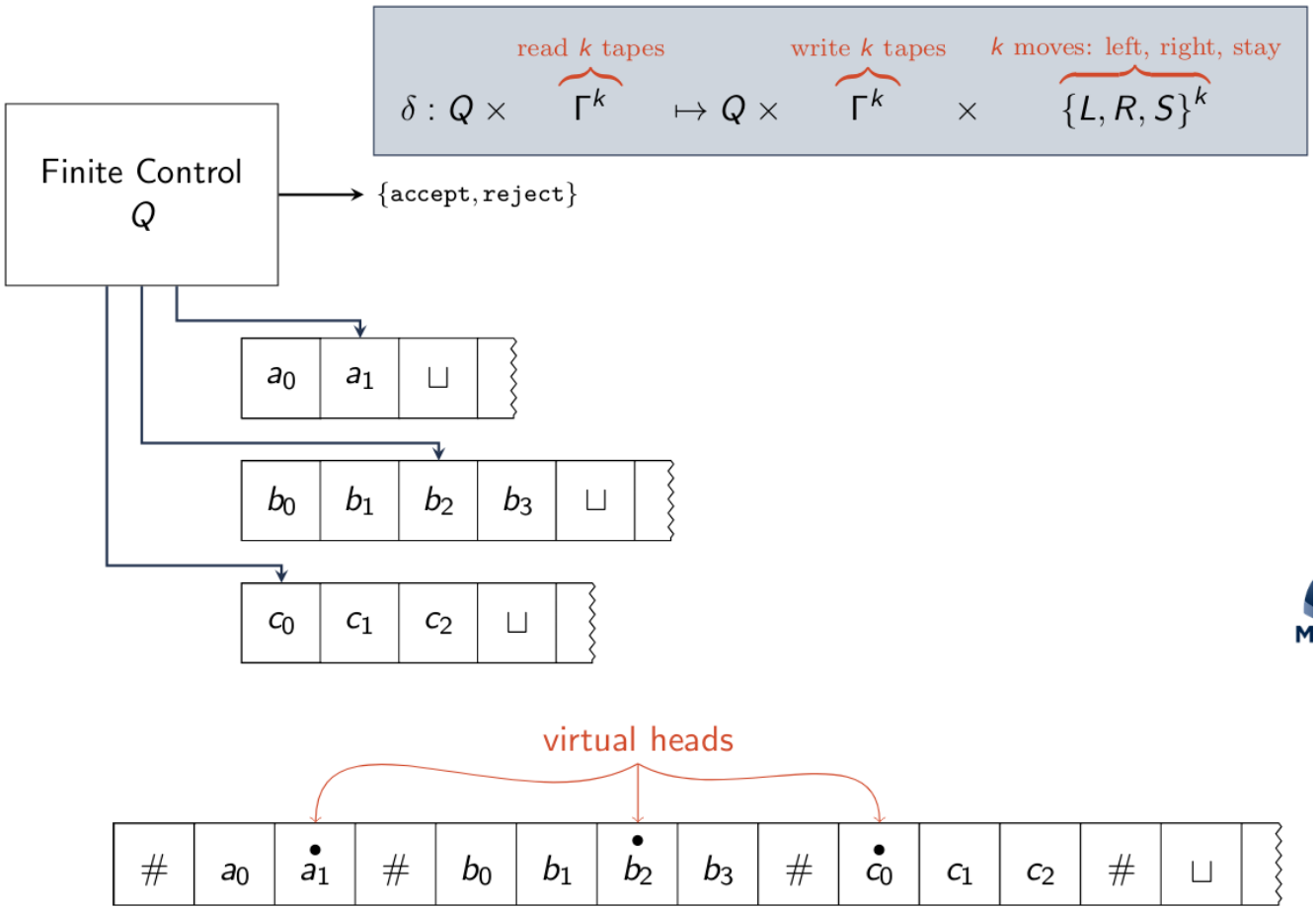
That is, $\exists \text{T}$, $L_{\text{acc}}(\text{T}) = \text{L}$

Language L is **Turing-recognizable** if there exists some Turing machine T that recognizes L.

That is, $\exists \text{T}$, $((L_{\text{acc}}(\text{T}) =) \wedge (L_{\text{rej}}(\text{T}) = \bar{\text{L}}))$

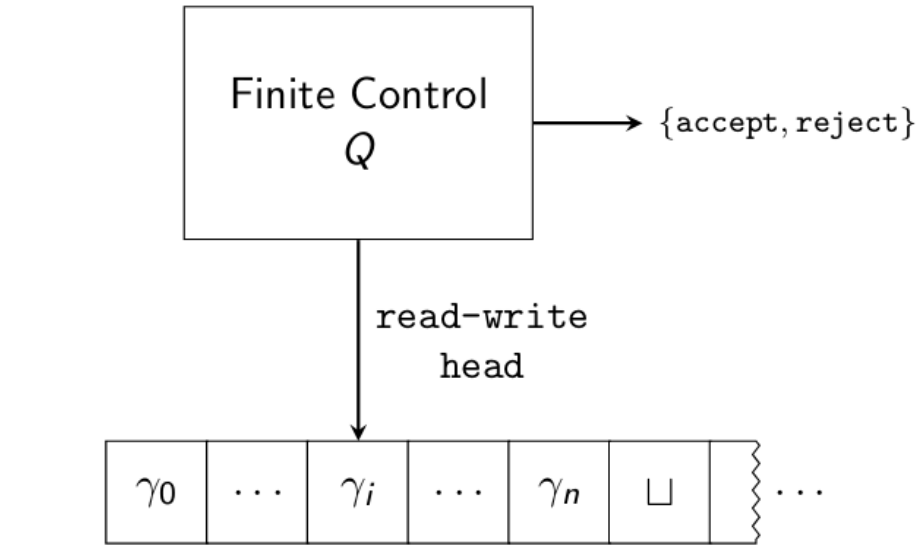
Variants of Turing Machines

Multi-Tape Turing Machines

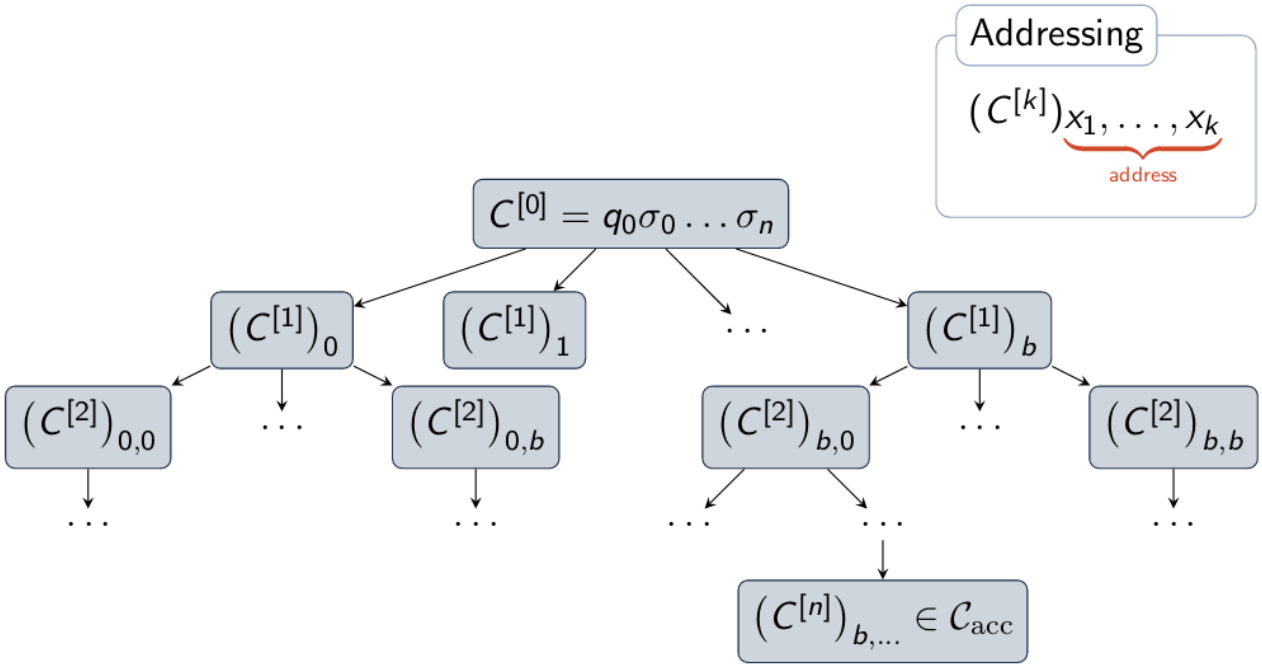


- To simulate a multi-tape move:
  1. Scan single-tape to determine symbols under each virtual head (finite permutations)
  2. Re-scan single-tape to update symbols and virtual head positions
- If a virtual head moves onto tape separator (#)
  1. Write a blank symbol (⊔) over the #.
  2. Shift tape contents right by one space.
  3. Resume simulation.

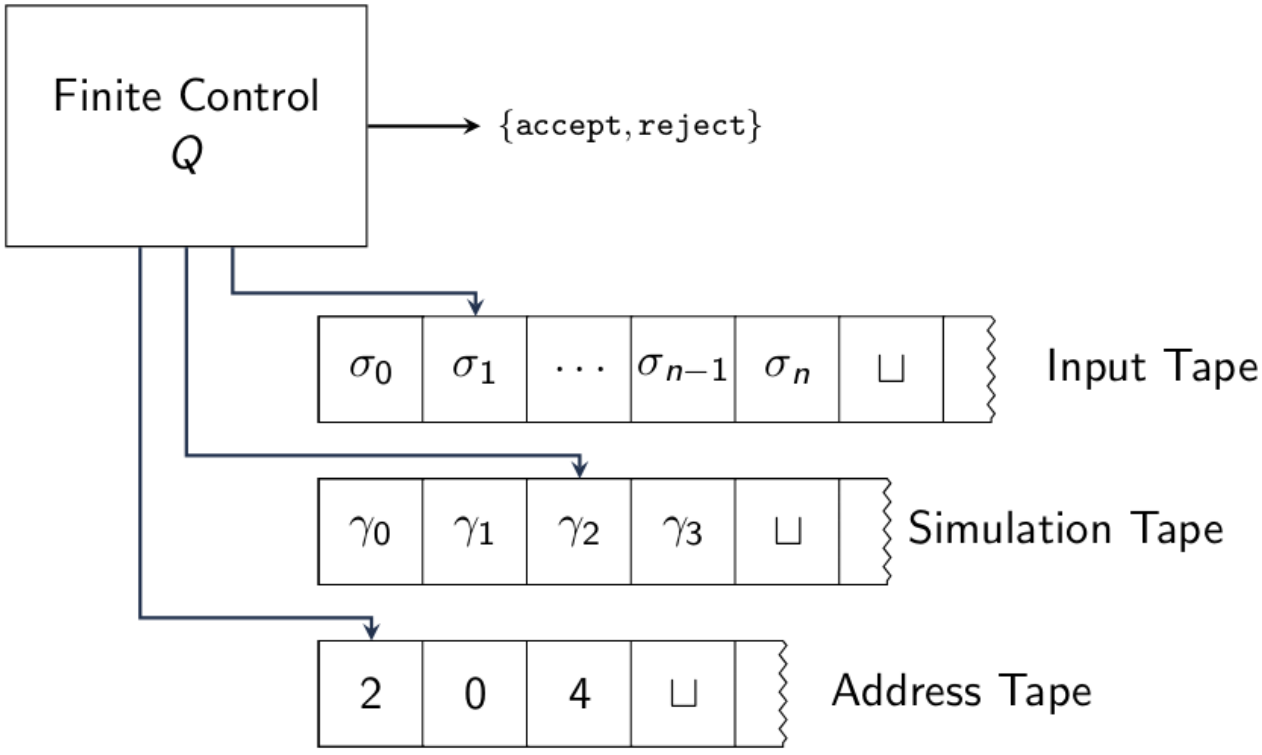
Nondeterministic Turing Machines



$$\delta : Q \times \Gamma \mapsto \underbrace{\mathcal{P}(Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}\})}_{\text{set of successor configurations}}$$



Deterministic Simulation: Iterative Deepening Search



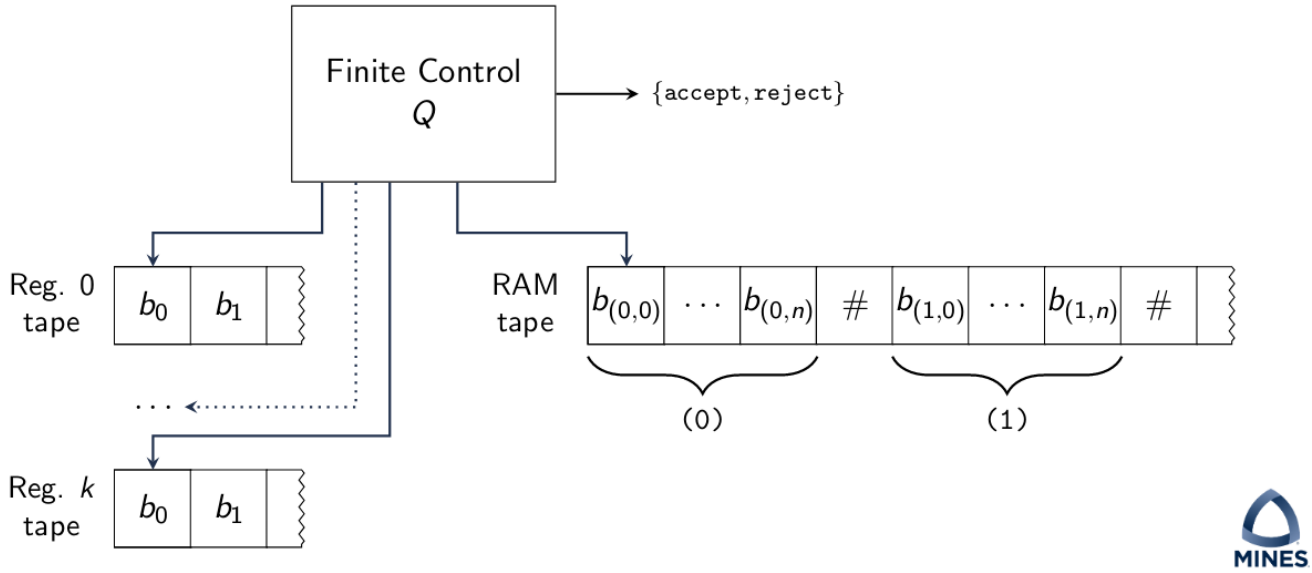
Deterministic Simulation Steps:

1. Initially, input tape contains input string σ (Simulation and Address tapes are empty)
2. Copy Input tape to Simulation tape
3. Use simulation tape to follow one branch of the nondeterminism

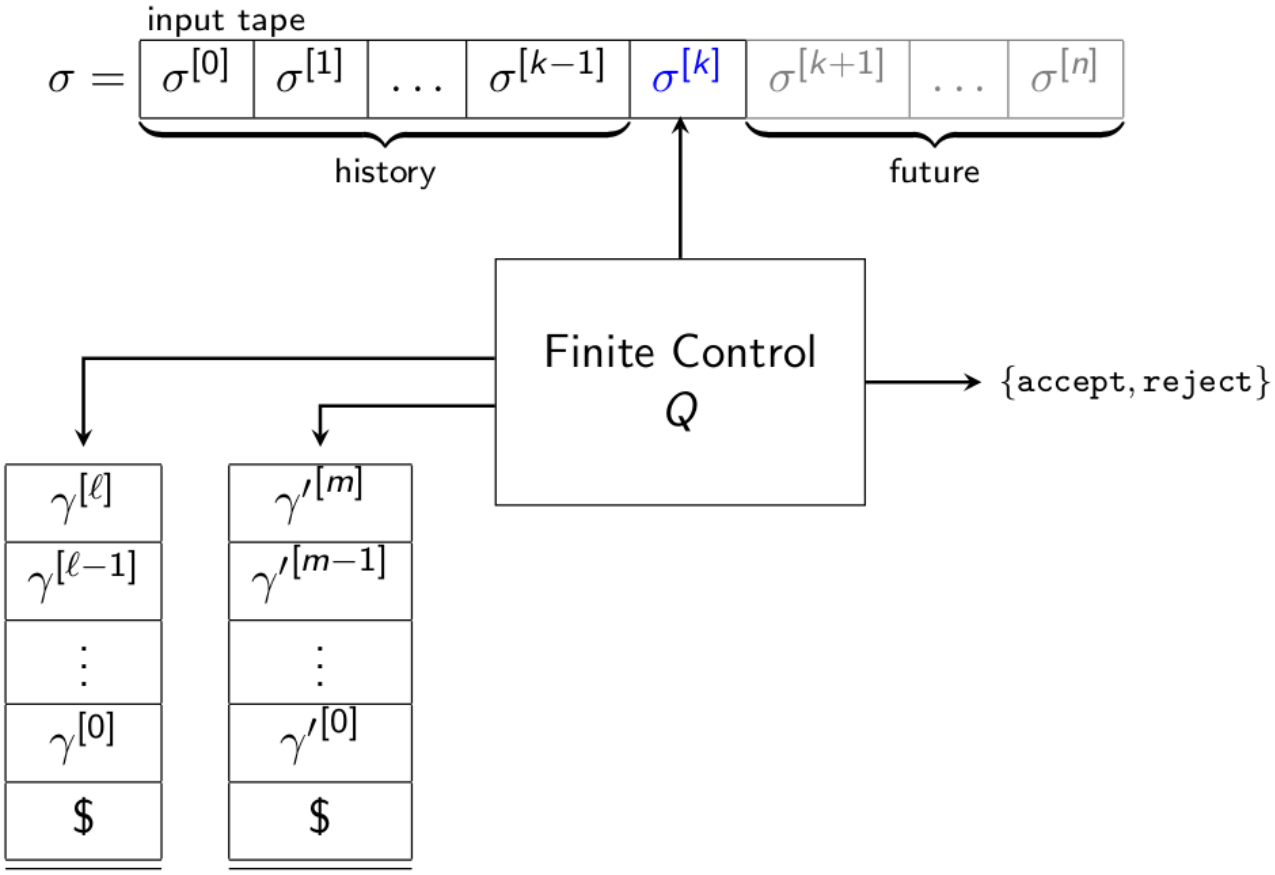
   3.1. Before each step, consult the address tape to determine which nondeterminism branch to follow

   3.2. If choice is invalid, no more address symbol remain (at current depth limit), or we reach a rejecting configuration, go to step 4
4. Increment the address (lexicographically next string). Go to step 2 to simulate the next branch or depth level.

RAM Machine



Two-Stack Pushdown Automata



Theorem: Two stack pushdown automata are Turing complete

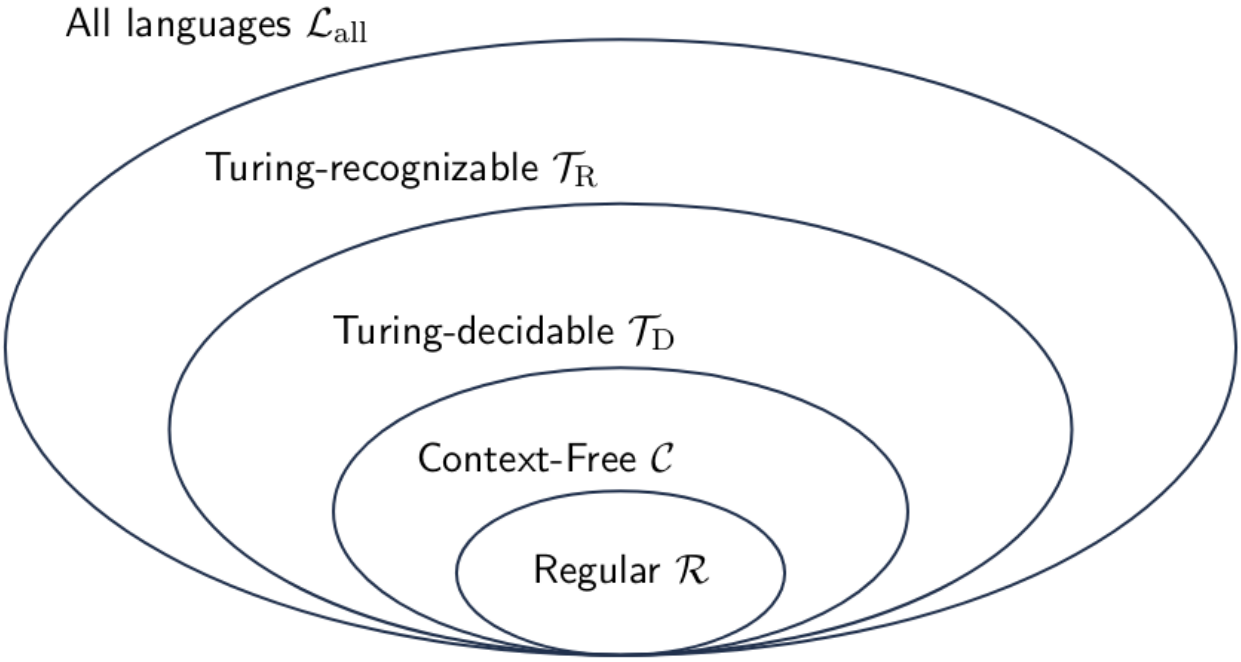
Proof outline:

1. Can construct Turing machine to simulate any two-stack PDA

2. Can construct two-stack PDA to simulate any Turing machine

Chomsky Hierarchy

$$\mathcal{R} \subset \mathcal{C} \subset \mathcal{T}_{\mathrm{D}} \subset \mathcal{T}_{\mathrm{R}} \subset \mathcal{L}_{\mathrm{all}}$$



Links

1. https://www.youtube.com/watch?v=D9eF_B8URnw
2. https://www.youtube.com/watch?v=cR4Re0YfoOo