Last Time

---

**Context-Free Languages (CFL)**

Intrinsinc: DFA + Stack

Non-deterministic Pushdown Automata (PDA)

---

Closure Properties

Then: $L_1$, $L_2$ are CFL

Then so are $L_1 \cup L_2$, $L_1 \cdot L_2$, $L_1^*$

- $S \rightarrow S_1 | S_2$
- $S \rightarrow SS_1 \epsilon$

Then: $L_1$ is CFL, $L_2$ is a regular lang.

Then $L_1 \cap L_2$ is CFL

(Product construction of a PDA + a DFA)

---

Rank: Not closed under intersection

**Example:**

$L_1 = \{0^i 1^i 2^k | i, k \geq 0 \}$ CFL

$L_2 = \{0^i 1^k 2^k | i, k \geq 0 \}$ CFL

$L_1 \cap L_2 = \{0^i 1^i 2^i | i \geq 0 \}$ is not a CFL.

Not closed under complements.
Turing Machine (TM) (early 1900s)

(How to define "all" that can be computed)

Turing (1936) - one of the simplest models. Resembles modern computers.

Main Features
- Infinite memory
- Can read & write both
- Can go back & forth in memory

Implementation
- Unbounded tape
- Read/write one symbol at current position ("head")
- Move head one position left/right

\[
\text{tape: } \epsilon x_1 x_2 \cdots x_i x_{i+1} \cdots
\]

Finite State Machine

deterministic

**Formal Def:** A TM is specified as

\[
M = (Q, \Sigma, \Gamma, \delta, q_0, B, q_{acc}, q_{ rej})
\]

where
- \(Q\) is a finite set of states
- \(\Sigma\) finite alphabets for input
- \(\Gamma\) finite "tape" alphabets \(\Sigma \subset \Gamma\)

\(q_{acc}\) accept state
\(q_{rej}\) reject state
1. **States**

- $q_0 \in \mathcal{Q}$ is the start state.
- $B \in \Gamma \setminus \{\varepsilon\}$ is the "blank" symbol.
- $\mathcal{F} \subseteq \mathcal{Q}$ is a unique accept state.
- $\mathcal{R} \subseteq \mathcal{Q}$ is a unique reject state.

2. **Transition Function**

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

- $q$ is the current state.
- $x$ is the read symbol.
- $q'$ is the next state.
- $a$ is the write symbol.
- $\ell$ is the move direction: left, right, stay.

3. **Configuration** (Instantaneous Configuration, or IC)

A configuration is a string $x_1 x_2 \ldots x_i q x_i \ldots x_n$ ($q \in \mathcal{Q}, x_i \in \Gamma$).

4. **Single Move**

Let $q \neq q_{\text{acc}}, q_{\text{rej}}$.

- If $\delta(q, a) = (q', a', S)$:
  $$q a B \xrightarrow{a} q' a' B$$
  $$a, B \in \Gamma^*$$

- Special case:
  $$q a \xrightarrow{a} q a' q_1$$

- If $\delta(q, a) = (q', a', R)$:
  $$q a B \xrightarrow{a} q' a B$$

- Special case:
  $$a a B \xrightarrow{R} \text{(crash)}$$
\[ S(q, a) = (q', a', L) \]
\[ \langle b \rangle q \beta \rightarrow^*_M \langle q' \rangle b \alpha \beta \]

**Def:** (Multiple Moves)
\[ C \sim^* C' \text{ if } C \sim^*_M C_1 \sim^*_M C_2 \ldots \sim^*_M C_k \sim^*_M C' \]

Where \( C, C_1, \ldots, C_k \) are some configurations.

**Def:** on input \( \alpha \in \Sigma^* \)
- \( M \) accepts \( x \) \( \iff \) \( q_0 x \sim^*_M \langle \beta \rangle \)
- \( M \) rejects \( x \) \( \iff \) \( q_0 x \sim^*_M \langle \gamma \rangle \)
- \( M \) crashes on \( x \) \( \iff \) \( q_0 x \sim^*_M \text{ Crash} \)
- \( M \) does not halt \( \iff \) \( q_0 x \sim^*_M \langle \gamma \rangle \sim^*_M \langle \gamma \rangle \sim^*_M \ldots \)

infinite seq.
\[ M \text{ does not halt on input } w \Rightarrow M \text{ infinite seq.} \]

\[ L(M) = \{ x \in \Sigma^* \mid M \text{ accepts } x \} \]

**Def:**
- \( L \) is recursively enumerable (r.e.)
  \( \iff L = L(M) \) for some TM \( M \).
- \( L \) is recursive (also called decidable)
  \( \iff L = L(M) \) for some TM \( M \) that halts on all inputs \( w \in \Sigma^* \).

**Rmk:**
**Diagram:**

\[ q_0 \xrightarrow{a} q_1 \]

**Notation**

\[ \delta(9, a) = (q_1, a', D) \]

\[ \delta(9, \lambda) = (q_{0a}, \lambda, S) \]

**Ex:**
\[ \{ 0^n1^n2^n \mid n \geq 0 \} \]

**Idea:** Repeat.

1. Change first 0 to x, go R until find 1.
1. Change first 0 to X, go R until find 1.
2. Change 1 to Y, go R until find Z.
3. Change Z to Z, go L until find X.

Until no more 0's.

Check no more 1's and 2's.

---

**Extensions:**
- Two-way infinite tape
- Multiple tapes
- Non-determinism
- Random access

- Tape is simple
  Can be simulated by a TM.
  (may take more time & space)

**Church-Turing Thesis:**

- Any language/function can be solved by some systematic procedure i.e. algorithm
- If it can be accepted/computed by a TM.

**Remark:**

"Thesis" (and not a Theorem) because can't be proved mathematically.

Think of it as "axiom/"def" of computable".

(So far no counter example! )