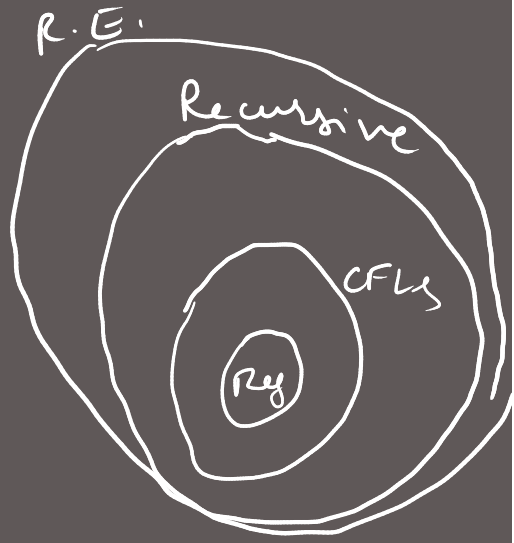
TURING MACHINES



DFA/NFA - regular languages

NPDA - Context free languages

DPDA - deterministic CFLs

Turing machines

- more powerful automata
- unrestricted memory.
- can compute anything that is "normally considered computable"

Early 1900's : David Hilbert posed some questions to the mathematics community.

→ Find an "effective procedure" to determine the solvability of Diophantine equations

→ Find an "effective procedure" to determine Truth/falsity of sentences in first-order Number theory.

Attempts to formalise "effective procedure"

- Turing machines (Turing)
- Post systems (Post)
- μ-recursive functions (Gödel)
- λ-calculus (Church)
- Markov Processes (Markov)

} work on different forms of data

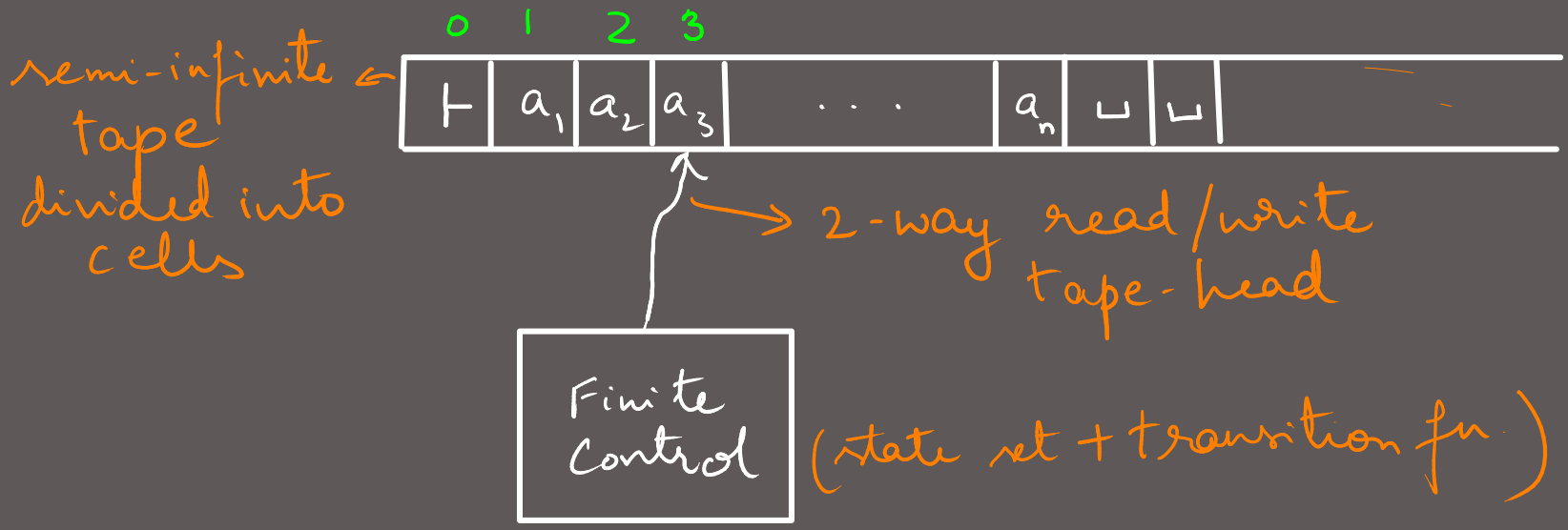
Church's Thesis (also called Church-Turing thesis)

All formalisms capture precisely our intuition of what is effectively computable.

Formal model of an effective procedure
- should have finite description
- consist of discrete steps that can be carried out mechanically.

Turing machines
- readily programmable
- computations are simple & mechanistic in nature.

Informal Description of a Turing Machine



Formal Description

A deterministic one-tape Turing machine is a 9-tuple

$$(Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t, r)$$

Q: finite set of states

Σ: input alphabet

Γ: tape alphabet ($\Sigma \subseteq \Gamma$)

$\vdash$: left endmarker $\vdash \in \Gamma \setminus \Sigma$

$\sqcup$: blank symbol $\sqcup \in \Gamma \setminus \Sigma$

$s \in Q$: start state

$t \in Q$: accept state

$r \in Q$: reject state

$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$: transition function

$\delta(p, a) = (q, b, R)$: Upon reading a while in state p, the TM enters state q, replaces a with b and moves one cell to the right

Restrict TM so that

→ left endmarker is never overwritten with another symbol & the machine never moves off the tape to the left of $\vdash$

$\forall p \in Q, \exists q \in Q$ s.t $\delta(p, \vdash) = (q, \vdash, R)$

→ Once the machine enters accept/reject states it never leaves.

$\forall b \in \Gamma, \exists c, c' \in \Gamma, dd' \in \{L, R\}$

$\delta(t, b) = (t, c, d)$, $\delta(r, b) = (r, c', d')$

Configuration of a TM

element of $Q \times \Gamma^* \times \mathbb{N}$ → $\{0, 1, 2, \ldots\}$

(p, γ, n) → position of tape head

- p ← current state
- γ ↓ current tape contents upto rightmost non-blank symbol

Start configuration upon input $x \in \Sigma^*$: $(s, \vdash x, 0)$

Next Configuration Relation $\left(\xrightarrow[M]{1}\right)$

γ_n : n^{th} symbol of string γ.

$s_b^n(\gamma)$: substituting γ_n with b at position n of γ.

$$(p, \gamma, n) \xrightarrow[M]{1} \begin{cases} (q, s_b^n(\gamma), n-1) & \text{if } \delta(p, \gamma_n) = (q, b, L) \\ (q, s_b^n(\gamma), n+1) & \text{if } \delta(p, \gamma_n) = (q, b, R) \end{cases}$$

Reflexive transitive Closure $\left(\xrightarrow[M]{*}\right)$

(defined inductively)

α, β, γ : configurations

- $\alpha \xrightarrow[M]{0} \alpha$
- $\alpha \xrightarrow[M]{n+1} \beta$ if $\alpha \xrightarrow[M]{n} \gamma$ & $\gamma \xrightarrow[M]{1} \beta$ for some γ
- $\alpha \xrightarrow[M]{*} \beta$ if $\alpha \xrightarrow[M]{n} \beta$ for some $n \geq 0$

Acceptance

M accepts $x \in \Sigma^*$ if

$(s, \vdash x, 0) \xrightarrow[M]{*} (t, \vdash y, n)$ for some $y \in \Gamma^*$ & $n \in \mathbb{N}$

Rejection

M rejects $x \in \Sigma^*$ if

$(s, \vdash x, 0) \xrightarrow[M]{*} (r, \vdash y, n)$ for some $y \in \Gamma^*$ & $n \in \mathbb{N}$.

More Definitions

- M halts if it either accepts or rejects.
- Language of M

$$L(M) = \{x \in \Sigma^* : M \text{ accepts } x\}$$

- $A \subseteq \Sigma^*$ is recursively enumerable if $A = L(M)$ for some Turing machine M.
- Turing machine M is total if it halts on all inputs
- $A \subseteq \Sigma^*$ is recursive if $A = L(M)$ for some total TM M.
- M loops if it does not halt

- $A \subseteq \Sigma^*$ is co-r.e. if $\neg A$ is r.e.

  $[\neg A = \Sigma^* \setminus A]$