Modeling Computation

Introduction to Formal Languages and Automata

Turing Machines
Finite Automata (DFAs, NFAs)
• Storage: None
• Recognize regular languages

Pushdown Automata (PDAs)
• Storage: Single stack
• Recognize context-free languages

What languages can be recognized by any computational device whatsoever?
A more powerful model of a computer

1928: Hilbert’s *Entscheidungsproblem.*
1936: Alan Turing proposes the *a-machine,* a model of any possible computation. Shows that a general solution to the *entscheidungsproblem* is impossible.
Turing Machine

Finite control

Read/Write head - moves left and right

Input/Output tape - infinite length
Turing Machine (TM)

\[ M = (Q, \Sigma, \Gamma, \delta, q_0, B, F) \]
\[ Q = \text{finite set of states} \]
\[ \Sigma = \text{finite input alphabet} \]
\[ \Gamma \supseteq \Sigma \text{ finite set of tape symbols} \]
\[ \delta: (Q \times F) \times \Gamma \rightarrow (Q \times \Gamma \times \{ \leftarrow, \rightarrow \}) \]
\[ q_0: \text{initial state } q_0 \in Q \]
\[ B \in \Gamma \ominus \Sigma \text{ is the blank symbol} \]
\[ F \subseteq Q \text{ is the set of final or accepting states. (halting)} \]
The language $L(M)$ of TM $M$ is the set of strings $w \in \Sigma^*$ such that $M$ halts on $w$.

The class of languages recognized by Turing machines is called **recursively enumerable**.
Graphical Representations of TMs

- **start state**
- **final state**

"a/X →" means:
- read "a" from tape
- write "X" to tape
- move head right

transitions:
- **B/B →**
- **b/b →**

B is blank
Let $M(w)^{\uparrow}$ denote that TM $M$ never halts on input $w$.

A configuration $C = (S, s, a, B)$ specifies:
- current state
- string to left of tape head
- symbol under tape head
- string to right of tape head
Let $C_1 \vdash C_2$ denote that configuration $C_1$ yields $C_2$ in one step by applying a transition $\delta(s, a), s \in Q, a \in \Sigma$.

$C_1 \vdash \cdots \vdash C_n$ denotes a sequence of transition in $C_1 \vdash C_n$ is a computation.
Suppose $M$ starts in $(q_0, w)$ with head over the leftmost symbol in $w$ and halts in $(h, u)$ for $h \in F$ and $u \in \Sigma^*$. 

\[(q_0, w) \xrightarrow{\#} (h, u)\]

Denote \[\_\] by $M(w)$ and call it the output of $M$ on $w$. 
Let $f: \Sigma^* \to \Sigma^*$ be a partial or total function.

Let $\text{Dom}(f) = \{w \mid f(w) \text{ is defined}\}$.

$M$ computes $f$ if $M$ halts on every $w \in \text{Dom}(f)$ with $M(w) = f(w)$ and $M(w) \uparrow$ on every $w \notin \text{Dom}(f)$. 
Partial Turing Computable function $f$

$\exists \text{Turing Machine } M \text{ that computes } f,$
and $\text{Dom}(f) = \Sigma^*$

Turing Computable function $f$

Partial Turing Computable, and
$\text{Dom}(f) = \Sigma^*$

Turing Computable language $L$: (decidable language)

Characteristic function is Turing Computable.

$$\eta_L(w) = \begin{cases} 1 & \text{if } w \in L \\ 0 & \text{otherwise} \end{cases}$$
Undecidable problems are those whose corresponding language is not Turing computable.

The Church-Turing Thesis:

Every effectively calculable function is computable.
Building Turing Machines

Build up large TMs using smaller ones like subroutines and define correct interfaces.

Example: Accept $w \in \{a^n b^n c^n \mid n > 0\}$

Strategy: (1) overwrite sets of $a$, $b$, $c$ with $XXX$ (first $a$, first $b$, first $c$ found)

(2) check that only $X$s remain
    if so, halt, else goto 1
Accept $w \in \{a^n b^n c^n \mid n > 0\}$

- **Start** → erase_abc → transition → check → transition → halt

**erase_abc:** find 1st “a”, replace w/ “X”, then find 1st “b”, replace w/ “X”, then find 1st “c”, replace w/ “X”, then return head to the front.

**check:** stream right through “X”s, if end at “a” return to the front.

**halt:** accept
erase_a:

erase_b:

erase_c:

goto_front:
check:

halt: