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

finite control

read/write head

... B B 1X_1X_2 ... X_n B ... }
Turing Machine (TM)

\[ M = (Q, \Sigma, \Gamma, \delta, q_0, B, F) \]

- **Q** = finite set of states
- **\Sigma** = finite input alphabet
- **\Gamma** = finite set of tape symbols
- **\delta** : \((Q - F) \times \Gamma\) \rightarrow (Q \times \Gamma \times \{\leftarrow, \rightarrow\})
- **q_0** : initial state \( q_0 \in Q \)
- **B** \in (\Gamma - \Sigma) is the blank symbol
- **F \subseteq Q** 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 \underline{recursively enumerable}. 
Graphical Representations of TMs

A graphical representation of a TM includes:
- **Start state** (q0)
- **Final state** (f)
- **Transitions** (e.g., b/b → b/b, a/X → B/B, B/B → B/B, B/B → B/B)

“a/X →” means:
- Read “a” from tape
- Write “X” to tape
- Move head right

B is blank

Replace “a”s with “X”s, leave “B”s alone.
Let $M(w) \uparrow$ denote that TM $M$ never halts on input $w$.

A configuration $C = (S, x, a, b)$ 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 $\delta(s, a), s \in Q, a \in \Sigma$.

$C_1 \vdash \ast C_n$ denotes $C_1 \vdash C_2 \vdash \cdots \vdash C_n$

Sequence of transition in $C_1 \vdash \ast C_n$ 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 $\cup$ by $M(w)$ and call it the output of $M$ on $w$. 
Let $f : \Sigma^* \rightarrow \Sigma^*$ 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 $f$

$\exists \text{TM } M \text{ that computes } f,$
and $\text{Dom}(f) \subseteq \Sigma^*$

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

no algorithm (TM that always halts) to solve the problem.

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 XXXX (first \( a \), first \( b \), first \( c \) found)

2. Check that only Xs 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: Stream through $X$s
ftrnds $U^\text{blank} \rightarrow \text{goto halt}$
ftrnds $a \rightarrow \text{goto front} \rightarrow \text{goto erase}_{abc}$

halt:
erase abc:

all missing transitions go to trap state

U/U → a/U → b/U → c/U → X/U →
Check:

\[ X/X \rightarrow a/a \leftarrow u/u \rightarrow \text{erase}_{abc} \]

\[ u/u \leftarrow \text{halt} \]

\[ X/X \leftarrow \]
Complete machine:

Put it all together

 implied by missing transitions
accept by characteristic function

\[ \text{halt: } \quad \begin{array}{c}
\quad \xrightarrow{U/L} \\
\quad \xleftarrow{X/U}
\end{array} \]

\[ \text{trap: } \quad \begin{array}{c}
\quad \xrightarrow{U/X} \\
\quad \xleftarrow{U/U} \\
\quad \xrightarrow{X/O}
\end{array} \]

\[ \xrightarrow{r/2} \quad \xrightarrow{r/2} \quad \xrightarrow{r/2} \quad \xrightarrow{U/U} \]
Verify \( aabbcc \in L \)

Verify \( ababcc \in L \)

\((q_0, aabbcc) \rightarrow (q_0, Xabbbcc) \rightarrow \cdots\)
Does the machine accept abcabc?