1 Unrestricted Computation

General Computing Machines

- Machines so far: DFAs, NFAs, PDAs
  - Limitations on how much memory they can use: fixed amount of memory plus (for PDAs) a stack
  - Limitations on what they can compute/decide: only regular languages or context free languages
- The complete machine?
  - No limitations on memory usage? And maybe other ways to use computational resources that we haven’t thought of...
    * Come up with a model that describes all “conceivable” computation
  - No limitation on what they can compute?
    * No! There are far too many languages over \( \{0,1\} \) than there are “machines” or programs (as long as machines can be represented digitally)

---

General Computing Machines

*Alonzo Church, Emil Post, and Alan Turing (1936)*

Figure 1: Alonzo Church

Figure 2: Emil Post
Church (\(\lambda\)-calculus), Post (Post’s machine), Turing (Turing machine) independently came up with formal definitions of mechanical computation

- All equivalent!
- In this course: Turing Machines

2 Turing Machines

2.1 Definition

Turing Machines

- Unrestricted memory: an infinite tape
  - A finite state machine that reads/writes symbols on the tape
  - Can read/write anywhere on the tape
  - Tape is infinite in one direction only (other variants possible)

- Initially, tape has input and the machine is reading (i.e., tape head is on) the leftmost input symbol.

- Transition (based on current state and symbol under head):
  - Change control state
  - Overwrite a new symbol on the tape cell under the head
Turing Machines

**Formal Definition**

A Turing machine is $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$ where

- $Q$ is a finite set of control states
- $\Sigma$ is a finite set of input symbols
- $\Gamma \supseteq \Sigma$ is a finite set of tape symbols. Also, a blank symbol $\sqcup \in \Gamma \setminus \Sigma$
- $q_0 \in Q$ is the initial state
- $q_{acc} \in Q$ is the accept state
- $q_{rej} \in Q$ is the reject state, where $q_{rej} \neq q_{acc}$
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function.

Given the current state and symbol being read, the transition function describes the next state, symbol to be written and direction (left or right) in which to move the tape head.

**Transition Function**

\[ \delta(q_1, X) = (q_2, Y, L) \]: Read transition as “the machine when in state $q_1$, and reading symbol $X$ under the tape head, will move to state $q_2$, overwrite $X$ with $Y$, and move its tape head to the left”

- In fact $\delta : (Q \setminus \{q_{acc}, q_{rej}\}) \times \Gamma \to Q \times \Gamma \times \{L, R\}$. No transition defined after reaching $q_{acc}$ or $q_{rej}$
- Transitions are deterministic
- Convention: if $\delta(q, X)$ is not explicitly specified, it is taken as leading to $q_{rej}$, i.e., say $\delta(q, X) = (q_{rej}, \sqcup, R)$

**Configurations**

The configuration (or “instantaneous description”) contains all the information to exactly capture the “current state of the computation”

\[ X_1 X_2 \cdots X_{i-1} q X_i \cdots X_n \]
• Includes the current state: \( q \)
• Position of the tape head: Scanning \( i \)th symbol \( X_i \)
• Contents of all the tape cells till the rightmost nonblank symbol. This is will always be finitely many cells. Those symbols are \( X_1 X_2 \cdots X_n \), where \( X_n \neq \sqcup \) unless the tape head is on it.

**Special Configurations**

• Start configuration: \( q_0 X_1 \cdots X_n \), where the input is \( X_1 \cdots X_n \)
• Accept and reject configurations: The state \( q \) is \( q_{\text{acc}} \) or \( q_{\text{rej}} \), respectively. These configurations are halting configurations, because there are no transitions possible from them.

**Single Step**

**Definition 1.** We say one configuration \((c_1)\) **yields** another \((c_2)\), denoted as \( c_1 \vdash c_2 \), if one of the following holds.

- If \( \delta(q, X_i) = (p, Y, \text{L}) \) then
  \[ X_1 X_2 \cdots X_{i-1} q X_i X_{i+1} \cdots X_n \vdash X_1 X_2 \cdots X_{i-2} p X_{i-1} Y X_{i+1} \cdots X_n \]
  Boundary Cases:
  - If \( i = 1 \) then \( q X_1 X_2 \cdots X_n \vdash p Y X_2 \cdots X_n \)
  - If \( i = n \) and \( Y = \sqcup \) then \( X_1 \cdots X_n q X_n \vdash X_1 \cdots p X_{n-1} \)

- If \( \delta(q, X_i) = (p, Y, \text{R}) \) then
  \[ X_1 X_2 \cdots X_{i-1} q X_i X_{i+1} \cdots X_n \vdash X_1 X_2 \cdots X_{i-2} Y p X_{i+1} \cdots X_n \]
  Boundary Case:
  - If \( i = n \) then \( X_1 \cdots X_{n-1} q X_n \vdash X_1 \cdots X_{n-1} p \sqcup \)

**Computations**

**Definition 2.** We say \( c_1 \models^* c_2 \) if the machine can move from \( c_1 \) to \( c_2 \) in zero or more steps, i.e., \( c_1 = c_2 \) or there exist \( c'_1, \ldots, c'_n \) such that \( c_1 = c'_1, c_2 = c'_n \) and \( c'_i \vdash c'_{i+1} \)

**Acceptance and Recognition**
**Definition 3.** A Turing machine $M$ accepts $w$ iff $q_0w \vdash^* \alpha_1 q_{\text{acc}} \alpha_2$, where $\alpha_1, \alpha_2$ are some strings. In other words, the machine $M$ when started in its initial state and with $w$ as input, reaches the accept state.

Note: The machine may not read all the symbols in $w$. It may pass back and forth over some symbols of $w$ several times. Finally, $w$ may have been completely overwritten.

**Definition 4.** For a Turing machine $M$, define $L(M) = \{w \mid M \text{ accepts } w\}$. $M$ is said to accept or recognize a language $L$ if $L = L(M)$.

### 2.2 Examples

**Example 1:** TM for $\{0^n1^n \mid n > 0\}$

Design a TM to accept the language $L_{0n1n} = \{0^n1^n \mid n > 0\}$

**High level description**

On input string $w$

while there are unmarked 0s, do

Mark the leftmost 0

Scan right till the leftmost unmarked 1;

if there is no such 1 then crash

Mark the leftmost 1

done

Check to see that there are no unmarked 1s;

if there are then crash

accept

---

**Example 1:** TM for $\{0^n1^n \mid n > 0\}$

```
\begin{figure}
\centering
\begin{tikzpicture}
\node[state, initial, accepting] (q0) at (0,0) {$q_0$};
\node[state, accepting] (q1) at (2,0) {$q_1$};
\node[state, initial] (q2) at (4,0) {$q_2$};
\node[state] (q3) at (0,-2) {$q_3$};
\node[state] (q5) at (4,-3) {$q_{\text{acc}}$};

\draw[->] (q0) -- node {$A \rightarrow A, R$} (q1);
\draw[->] (q1) -- node {$1 \rightarrow B, L$} (q2);
\draw[->] (q0) -- node {$B \rightarrow B, R$} (q3);
\draw[->] (q3) -- node {$\square \rightarrow \square, R$} (q5);
\draw[->] (q2) -- node {$B \rightarrow B, L$} (q5);
\draw[->] (q3) -- node {$0 \rightarrow 0, R$} (q2);
\draw[->] (q3) -- node {$B \rightarrow B, R$} (q5);
\draw[->] (q5) -- node {$0 \rightarrow 0, L$} (q2);
\end{tikzpicture}
\end{figure}
```
• Accepts input 0011:
  \[ q_00011 \vdash Aq_1011 \vdash A0q_11 \vdash Aq_20B1 \vdash q_2A0B1 \vdash q_00B1 \vdash AAq_1B1 \vdash AABq_1 \vdash AAq_2BB \vdash Aq_2ABB \vdash A \]

• Rejects input 00:
  \[ q_000 \vdash Aq_10 \vdash A0q_1 \vdash A0 \vdash q_{rej} \]

---

**Example:** \( \{0^n1^n \mid n > 0\} \)

**Formal Definition**

The machine is \( M = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej}) \) where

- \( Q = \{q_0, q_1, q_2, q_3, q_{acc}, q_{rej}\} \)
- \( \Sigma = \{0, 1\} \), and \( \Gamma = \{0, 1, A, B, \underline{\bot}\} \)
- \( \delta \) is given as follows

\[
\begin{align*}
\delta(q_0, 0) &= (q_1, A, R) & \delta(q_0, B) &= (q_3, B, R) \\
\delta(q_1, 0) &= (q_1, 0, R) & \delta(q_1, B) &= (q_1, B, R) \\
\delta(q_1, 1) &= (q_2, B, L) & \delta(q_2, B) &= (q_2, B, L) \\
\delta(q_2, 0) &= (q_2, 0, L) & \delta(q_2, A) &= (q_0, A, R) \\
\delta(q_3, B) &= (q_3, B, R) & \delta(q_3, \underline{\bot}) &= (q_{acc}, \underline{\bot}, R)
\end{align*}
\]

In all other cases, \( \delta(q, X) = (q_{rej}, \underline{\bot}, R) \). So for example, \( \delta(q_0, 1) = (q_{rej}, \underline{\bot}, R) \).

---

**Example 2:** TM for \( \{a^n b^n c^n \mid n > 0\} \)

Design a TM to accept the language \( L_{anbncn} = \{a^n b^n c^n \mid n > 0\} \)

**High level description**

On input string \( w \)
while there are unmarked as, do
  Mark the left most a
  Scan right to reach the leftmost unmarked b;
    if there is no such b then crash
  Mark the leftmost b
  Scan right to reach the leftmost unmarked c;
    if there is no such c then crash
  Mark the leftmost c
done
Check to see that there are no unmarked cs or cs;
  if there are then crash
accept
Example 2: TM for \{a^n b^n c^n \mid n > 0\}

Deciding a Language

- Only halting configurations are those with state $q_{\text{acc}}$ or $q_{\text{rej}}$
- A Turing machine may keep running forever on some input
- Then the machine does not accept that input
- So two ways to not accept: reject or never halt

**Definition 5.** A Turing machine $M$ is said to *decide* a language $L$ if $L = L(M)$ and $M$ halts on every input

Deciding a language is more than recognizing it. There are languages which are recognizable, but not decidable.