LECTURE - 9

"TURING MACHINES"

Regular Languages
    DFA/NFAs

- string
- concat
- choice
- looping

$0^*1^n$ is not regular, but is context free.

Context free languages
    - recursion

$0^*1^2$

Turing Machine
    FSM with access to unrestricted memory

FSM
Turing Machine

\[ \delta : (Q \setminus \{ \text{accept}, \text{reject} \}) \times \Gamma \to Q \times \Gamma \times \mathbb{Z} \]

Configuration \( (q, x, i) \in Q \times \Gamma^* \times \mathbb{N} \)

\( (p, x, i) \Rightarrow (q, y, j) \)

\( \delta(p, a) = (q, b, +1) \)

\( (p, xay, i) \Rightarrow (q, xby, i+1) \)
RAM: Random Access Memory.

Any language decidable in time $T(n)$ by a RAM is decidable in time $(T(n))^2$ by TM.

**VARIATIONS.**

- many accepting/rejecting states
- either write or move head
- doubly infinite tape
- many heads
- insert and delete cells
- multiple tapes each with their own head
- random access.

---

**Hilbert:**

Want an algorithm, s.t. if I feed a theorem, it will tell me whether theorem is true or false.

**Alan Turing:** Such an algorithm cannot exist. (Also Gödel's incompleteness theorems).
TURING MACHINES

Input to universal T.M. is \((M', x)\)

Input to \(M'\) is the string \(x\).

Can feed a turing machine \(M\) its own description as input.

Suppose there existed a machine for \(HALT\).
Then I can use this machine to build:

```
DOESNOTHALT
```

![Diagram](image)

**What does DOESNOTHALT do on input M' = DOESNOTHALT?**

- Suppose it outputs YES.
  - This means DOESNOTHALT halted and output YES on input DOESNOTHALT.
  - But it should only do this if M' = DOESNOTHALT does not halt on input M'. Thus a CONTRADICTION.

- Suppose it loops forever.
  - This means DOESNOTHALT doesn't halt on input DOESNOTHALT.
  - M' on input M' (= DOESNOTHALT) does not halt.
  - It should've halted and output YES. Thus a CONTRADICTION.

TURING MACHINES CANNOT DECIDE WHETHER A T.M. HALTS OR NOT.
<table>
<thead>
<tr>
<th>State Transition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \delta( p, a) = ( q, b, \delta) )</td>
<td>explanation</td>
</tr>
<tr>
<td>( \delta(\text{start}, \emptyset) = (\text{seek1}, $, +1) )</td>
<td>mark first ( \emptyset ) and scan right</td>
</tr>
<tr>
<td>( \delta(\text{start}, x) = (\text{verify}, $, +1) )</td>
<td>looks like we're done, but let's make sure</td>
</tr>
<tr>
<td>( \delta(\text{seek1}, \emptyset) = (\text{seek1}, \emptyset, +1) )</td>
<td>scan rightward for 1</td>
</tr>
<tr>
<td>( \delta(\text{seek1}, x) = (\text{seek1}, x, +1) )</td>
<td>mark 1 and continue right</td>
</tr>
<tr>
<td>( \delta(\text{seek1}, 1) = (\text{seek0}, x, +1) )</td>
<td>scan rightward for 0</td>
</tr>
<tr>
<td>( \delta(\text{seek0}, 1) = (\text{seek0}, 1, +1) )</td>
<td>mark ( \emptyset ) and scan left</td>
</tr>
<tr>
<td>( \delta(\text{seek0}, \emptyset) = (\text{reset}, x, +1) )</td>
<td>scan leftward for $</td>
</tr>
<tr>
<td>( \delta(\text{reset}, \emptyset) = (\text{reset}, \emptyset, -1) )</td>
<td>step right and start over</td>
</tr>
<tr>
<td>( \delta(\text{reset}, 1) = (\text{reset}, 1, -1) )</td>
<td>scan right for any unmarked symbol</td>
</tr>
<tr>
<td>( \delta(\text{reset}, x) = (\text{reset}, x, -1) )</td>
<td>success!</td>
</tr>
<tr>
<td>( \delta(\text{verify}, x) = (\text{verify}, x, +1) )</td>
<td>success!</td>
</tr>
</tbody>
</table>

The transition function for a Turing machine that decides the language \( \{0^n1^n \emptyset^n \mid n \geq 0\} \).
\[ \delta(\quad , a) = (\quad , b, \delta) \]
\[
\begin{align*}
\delta(\text{start}, \emptyset) &= (\text{seek1}, \$, +1) \\
\delta(\text{start}, x) &= (\text{verify}, \$, +1) \\
\delta(\text{seek1}, \emptyset) &= (\text{seek1}, \$, +1) \\
\delta(\text{seek1}, x) &= (\text{seek1}, x, +1) \\
\delta(\text{seek1}, 1) &= (\text{seek0}, x, +1) \\
\delta(\text{seek0}, 1) &= (\text{seek0}, 1, +1) \\
\delta(\text{seek0}, x) &= (\text{seek0}, x, +1) \\
\delta(\text{seek0}, 0) &= (\text{reset}, x, +1) \\
\delta(\text{reset}, 0) &= (\text{reset}, 0, -1) \\
\delta(\text{reset}, 1) &= (\text{reset}, 1, -1) \\
\delta(\text{reset}, x) &= (\text{reset}, x, -1) \\
\delta(\text{reset}, \$) &= (\text{start}, \$, +1) \\
\delta(\text{verify}, x) &= (\text{verify}, x, +1) \\
\delta(\text{verify}, \emptyset) &= (\text{accept}, \emptyset, -1)
\end{align*}
\]
\[
\begin{align*}
\Rightarrow (\text{seek1}, \$01\text{xx}0) \\
\Rightarrow (\text{seek1}, \$01\text{xx}0) \\
\Rightarrow (\text{seek0}, \$0\text{xx}0) \\
\Rightarrow (\text{seek0}, \$0\text{xx}0) \\
\Rightarrow (\text{reset}, \$0\text{xx}0) \\
\Rightarrow (\text{reset}, \$0\text{xx}0) \\
\Rightarrow (\text{reset}, \$0\text{xx}0) \\
\Rightarrow (\text{start}, \$0\text{xx}0) \\
\Rightarrow (\text{seek1}, \$\text{xx}0) \\
\Rightarrow (\text{seek1}, \$\text{xx}0) \\
\Rightarrow (\text{seek1}, \$\text{xx}0) \\
\Rightarrow (\text{seek1}, \$\text{xx}0) \\
\Rightarrow \text{reject!}
\end{align*}
\]
\[
\begin{align*}
\Rightarrow (\text{start}, \$01\text{xx}0) \\
\Rightarrow (\text{seek1}, \$01\text{xx}0) \\
\Rightarrow (\text{seek1}, \$01\text{xx}0) \\
\Rightarrow (\text{seek0}, \$0\text{xx}0) \\
\Rightarrow (\text{reset}, \$0\text{xx}0) \\
\Rightarrow (\text{reset}, \$0\text{xx}0) \\
\Rightarrow (\text{reset}, \$0\text{xx}0) \\
\Rightarrow (\text{start}, \$0\text{xx}0) \\
\Rightarrow (\text{seek1}, \$\text{xx}0) \\
\Rightarrow (\text{seek1}, \$\text{xx}0) \\
\Rightarrow (\text{seek1}, \$\text{xx}0) \\
\Rightarrow (\text{seek1}, \$\text{xx}0) \\
\Rightarrow \text{accept!}
\end{align*}
\]