Turing Machine Representation

1. List all the entries in the \( \delta \)-function (as in previous examples) but
   • cumbersome for more than 2 or 3 states
   • difficult to see the structure or pattern in the algorithm

2. Draw a **State Diagram** -
   states are represented by nodes
   rectangles for halting states
   circles for other states
   \( \delta \)-function entries as arrows from current state to new state

\[ \begin{align*}
q_0 &\xrightarrow{(a, b, d)} q_1 \\
\text{triple from } &\sum x \sum x \{-1, 0, 1\} \\
\text{symbol read} &\quad \text{symbol to write} \\
\text{head move} &
\end{align*} \]
for example, the odd/even Turing Machine

\[ \delta (q_0, 1) = (q_1, ^, 1) \]
\[ \delta (q_0, ^) = (q_2, ^, 0) \]
\[ \delta (q_1, 1) = (q_0, 1, 1) \]

If \( \delta(q, a) = (q', b, d) \) the graph has an arrow labeled \((a, b, d)\) from \(q\) to \(q'\)
3. as **Pseudocode**

   basic operations:  
   - TM read
   - TM write
   - head movement

   + control structures:  
   - if..then..else
   - while..do

   pseudocode feels like “proper programming” but ..

   it is easy to work at too high a level and to forget that
   - the TM cannot address its storage (tape), just read the symbol on the current-square
   - it cannot do additional operations ”on the side” eg counting,
an example: the **Head** function: build TM, M such that
\[ f_M(w) = \text{head}(w), \ w \in I^* \]
Input: a word w of I*, starting at square 0
Output: a word consisting of the first symbol of w, followed by ^.

\[ \Sigma = I \cup \{^\} \]
set of states, \(Q = \{\text{skip, erase, stop}\}\)

**δ-function:**
\[ \delta(\text{skip}, x) = (\text{erase}, x, 1), \ \text{all } x \in I \]
\[ \delta(\text{erase}, x) = (\text{stop}, ^, 0), \ \text{all } x \in I \]
so \(M = \{Q, \Sigma, I, \text{skip, erase, stop}\}\)

---

**state diagram**
**pseudocode:**

read symbol from current square
if symbol = ‘^’ then Halt and Fail
move right
write ‘^’ (whatever symbol is read)
Halt and Succeed.

eg.

```
C O M P U ^
```

**Input** - the string of symbols up to but not including the first ‘^’

**Output** - the string of symbols up to but not including the first ‘^’
..rest of tape ignored
pseudocode:

read symbol from current square
if symbol = ‘^’ then Halt and Fail
move right
write ‘^’ (whatever symbol is read)
Halt and Succeed.

eg.

\[
\begin{array}{cccccc}
C & ^ & M & P & U & ^ \\
\end{array}
\]

Input - the string of symbols up to but not including the first ‘^’

Output - the string of symbols up to but not including the first ‘^’
..rest of tape ignored
Example - Unary addition

Unary notation: represent $n$ by $1111..11$ ($n$ 1s)..written as $1^n$
Design TM, $M$, such that $f_M(1^n+. 1^m) = 1^{n+m}$

$I = \{1,+\}$
$\Sigma = \{1, +, ^\}$

```
    1   1   1   1   +   1   1   1   1   ^   ^
```

```
    1   1   1   1   1   1   1   1   ^   ^   ^..^
```
Unary addition - pseudocode

empty input?

if current symbol = ^ then Halt & Fail

while current-symbol = 1
    write 1
    move right

endwhile

if current-symbol = +
    write 1
    move right
else Halt & Fail
endif

while current-symbol = 1
    write 1
    move right
end while

if current-symbol = ^
    write ^
    move left
else Halt & Fail
endif

write ^
move left

else Halt & Fail
end if

write ^

Halt & Succeed
**Unary addition** - state diagram:

- **States:**
  - $q_0$
  - $q_1$
  - $q_2$
  - $q_3$

- **Transition function $\delta$:**
  - $\delta(q_0, 1) = (q_0, 1, 1)$
  - $\delta(q_0, +) = (q_1, 1, 1)$
  - $\delta(q_1, 1) = (q_1, 1, 1)$
  - $\delta(q_1, ^) = (q_2, ^, -1)$
  - $\delta(q_2, 1) = (q_3, ^, 0)$
  - $\delta(q_1, 1) = (q_1, 1, 1)$

- **Input Symbols:**
  - 1
  - +
  - ^

- **States Set $Q$:**
  - $\{q_0, q_1, q_2, q_3\}$

- **Final States $F$:**
  - $\{q_3\}$

**Diagram:**

- $q_0$ transitions to $q_0$ upon $1$.
- $q_0$ transitions to $q_1$ upon $+$.
- $q_1$ transitions to $q_1$ upon $1$.
- $q_1$ transitions to $q_2$ upon $^$.
- $q_2$ transitions to $q_3$ upon $1$.
- $q_3$ is the final state.
Unary addition - state diagram:

\[ Q = \{q_0, q_1, q_2, q_3\} \]
\[ F = \{q_3\} \]

\[ \delta\text{-function:} \]
\[ \delta(q_0, 1) = (q_0, 1, 1) \]
\[ \delta(q_0, +) = (q_1, 1, 1) \]
\[ \delta(q_1, 1) = (q_1, 1, 1) \]
\[ \delta(q_1, ^) = (q_2, ^, -1) \]
\[ \delta(q_2, 1) = (q_3, ^, 0) \]
**Unary addition - state diagram:**

- States: $Q = \{q_0, q_1, q_2, q_3\}$
- Final state: $F = \{q_3\}$

**δ-function:**
- $\delta(q_0, 1) = (q_0, 1, 1)$
- $\delta(q_0, +) = (q_1, 1, 1)$
- $\delta(q_1, 1) = (q_1, 1, 1)$
- $\delta(q_1, ^) = (q_2, ^, -1)$
- $\delta(q_2, 1) = (q_3, ^, 0)$
**Unary addition - state diagram:**

- **States:**
  - $q_0$
  - $q_1$
  - $q_2$
  - $q_3$

- **Transitions:**
  - $(+, 1, 1) ightarrow (1, 1, 1)$
  - $(^, ^, 1) ightarrow (q_2, ^, 0)$
  - $(+, ^, -1) ightarrow (q_0, 1, 1)$
  - $(1, ^, 0) ightarrow (q_1, 1, 1)$

- **Initial State:** $q_0$
- **Final State:** $q_3$
- **States:** $Q = \{q_0, q_1, q_2, q_3\}$
- **Final States:** $F = \{q_3\}$

**δ-function:**

- $\delta(q_0, 1) = (q_0, 1, 1)$
- $\delta(q_0, +) = (q_1, 1, 1)$
- $\delta(q_1, 1) = (q_1, 1, 1)$
- $\delta(q_1, ^) = (q_2, ^, -1)$
- $\delta(q_2, 1) = (q_3, ^, 0)$

**Input:**

```
1 1 1 1 1 1 ^
```
**Unary addition - state diagram:**

- **States:** $Q = \{q_0, q_1, q_2, q_3\}$
  - $F = \{q_3\}$

- **δ-function:**
  - $\delta(q_0, 1) = (q_0, 1, 1)$
  - $\delta(q_0, +) = (q_1, 1, 1)$
  - $\delta(q_1, 1) = (q_1, 1, 1)$
  - $\delta(q_1, ^) = (q_2, ^, -1)$
  - $\delta(q_2, 1) = (q_3, ^, 0)$
  - $\delta(q_3, 1) = (q_3, 1, 1)$

- **Transitions:**
  - $q_0 
  \begin{cases} 
    (+, 1, 1) & \text{on input } 1 \\
    (^, ^, -1) & \text{on input } ^ \\
  \end{cases}$

  - $q_1 
  \begin{cases} 
    (+, 1, 1) & \text{on input } 1 \\
    (^, ^, -1) & \text{on input } ^ \\
  \end{cases}$

  - $q_2 
  \begin{cases} 
    (+, 1, 1) & \text{on input } 1 \\
    (^, ^, -1) & \text{on input } ^ \\
  \end{cases}$

  - $q_3 
  \begin{cases} 
    (+, 1, 1) & \text{on input } 1 \\
    (^, ^, -1) & \text{on input } ^ \\
  \end{cases}$

- **Initial state:** $q_0$

- **Final state:** $q_3$
**Unary addition - state diagram:**

- **States:** $Q = \{q_0, q_1, q_2, q_3\}$
- **Final state:** $F = \{q_3\}$
- **δ-function:**
  - $\delta(q_0, 1) = (q_0, 1, 1)$
  - $\delta(q_0, +) = (q_1, 1, 1)$
  - $\delta(q_1, 1) = (q_1, 1, 1)$
  - $\delta(q_1, ^) = (q_2, ^, -1)$
  - $\delta(q_2, 1) = (q_3, ^, 0)$
  - $\delta(q_1, ^) = (q_2, ^, -1)$
  - $\delta(q_2, 1) = (q_3, ^, 0)$
Unary addition - state diagram:

- State diagram:
  - States: $Q = \{q_0, q_1, q_2, q_3\}$
  - Final state: $F = \{q_3\}$

- Delta function:
  - $\delta(q_0, 1) = (q_0, 1, 1)$
  - $\delta(q_0, +) = (q_1, 1, 1)$
  - $\delta(q_1, 1) = (q_1, 1, 1)$
  - $\delta(q_1, ^) = (q_2, ^, -1)$
  - $\delta(q_2, 1) = (q_3, ^, 0)$
Design a Turing Machine to implement the Tail function

Q = \{q_0, \ldots\}, \ F = \{\}, \ \sum = \{a, b, ^\}.

| a | b | a | b | ^ | ^ | ^ | ^ | ^ |

Method:
- if current symbol = ^, then move left. (gives Halt & Fail where input word = ε)
- else leave current symbol unchanged
- move right
- repeatedly {
  - Let current symbol be ‘s’; leave s unchanged and move left
  - write ‘s’
  - move right
  - if current-symbol = ^ then Halt & Succeed
  - else move right}
Design a Turing Machine to implement the Tail function

\[ Q = \{ q_0, q_1, q_2, q_3, q_4, q_5, q_6 \}, \ F = \{ q_6 \}, \ \Sigma = \{ a, b, ^ \}. \]

Algorithm 1 - “oscillating”
Design a Turing Machine to implement the Tail function

\[ Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}, \ F = \{q_5\}, \ \Sigma = \{a, b, ^\}. \]

Algorithm 2 - “shifting”

| a | b | a | b | \^ | \^ | \^ | \^ | \^ |

Method 2:
- If current symbol = \^ Halt & Fail.
- Repeat {move right} until current symbol = \^.
- Move left;
- Let ‘s’ be current symbol; write \^.
- Repeatedly: {move left write ‘s’, let ‘s’ be current symbol ; until reach square 0}**
- Halt & Succeed

** the **TM squares are not marked or individually addressable**, but the machine must be able to recognise when, moving left, it has reached square 0. This is done to avoid causing Halt & Fail by a further move left.

One method is to write a special symbol in square 0 which can be recognised later - often ^ can be used.
Design a Turing Machine to implement the Tail function

\[ Q = \{ q_0, q_1, q_2, q_3, q_4, q_5 \}, \ F = \{ q_5 \}, \sum = \{ a, b, ^ \}. \]

Algorithm 2 - “shifting”

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>a</th>
<th>b</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>a</td>
<td>b</td>
<td>^</td>
<td>^</td>
<td>^</td>
<td>^</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\delta(q_0, a) &= (q_1, ^, 1) \\
\delta(q_0, b) &= (q_1, ^, 1) \\
\delta(q_1, a) &= (q_1, a, 1) \\
\delta(q_1, b) &= (q_1, b, 1) \\
\delta(q_1, ^) &= (q_2, ^, -1) \\
\delta(q_2, ^) &= (q_5, ^, 0) \\
\delta(q_2, a) &= (q_3, ^, -1) \\
\delta(q_2, b) &= (q_4, ^, -1) \\
\delta(q_3, a) &= (q_3, a, -1) \\
\delta(q_3, b) &= (q_4, a, -1) \\
\delta(q_3, ^) &= (q_5, a, 0) \\
\delta(q_4, a) &= (q_3, b, -1) \\
\delta(q_4, b) &= (q_4, b, -1) \\
\delta(q_4, ^) &= (q_5, b, 0)
\end{align*}
\]
Summary

We have seen \( \delta \)-function
state diagram
pseudo-code
representations for Turing Machines and examples of these
for odd/even
Head
unary addition
Tail functions.

State diagrams permit visualisation of the TM and the structure
of an algorithm. They correspond directly to the \( \delta \)-function
representation.