Transitions

\[ S: \Sigma \times \Gamma \rightarrow \Sigma \times \Gamma \times \{L, R\} \]

Thus \( S(q, x) = (p, y, l) \) means that if in state \( q \) and tape head is scanning the symbol \( x \), then move to state \( p \), replace \( x \) by \( y \) on tape cell, and move tape head \( 1 \) cell left.

Deterministic TM (DTM), above defines a DTM — for each \( S(q, x) \) we have at most one possible move — although \( S(q, x) \) could be undefined.

Acceptance? If when DTM is started with \( w \) on tape it eventually enters a final state

Thus we may as well assume that all final/accepting states are “halting states” — in that no transitions are defined out of them.

Rejection? Halt in non-final state

\[ \text{NEVER HALT} \text{ (INFINITE LOOP)} \]

Recall in NFA/PDA we would halt when end of input is reached — then we check if state is final.

In T.M. input is not streaming by but is instead given on tape — so we need explicit notion of halting.

Halting when in state \( q \) and head sees \( x \), such that \( S(q, x) \) is undefined.

Any states have absolutely no transitions — always halts
EXAMPLE \( L = \{0^n1^n \mid n \geq 1 \} \)

INITIALLY
\[ \cdots \text{BBB} \text{OOOO}1111 \text{BBB} \cdots \]
\[ \uparrow \text{HEAD} \]

T.M. IDEA

MATCH LEFTMOST 0 WITH LEFTMOST 1,
REPLACING THEM BY \( X \) AND \( Y \) (RESPECTIVELY)
AND REPEAT.

T.M. M
\[ M = (Q, \Sigma, \Gamma, S, q_0, B, F) \]
\[ Q = \{q_0, q_1, q_2, q_3, q_4\} \quad F = \{q_4\} \]
\[ \Sigma = \{0, 1, ?\} \quad \Gamma = \{0, 1, X, Y, 1, B\} \]

TRANSITIONS

\[ \delta(q_0, 0) = (q_1, X, R) \quad \text{REPLACE 0 BY X AND LOOK} \]
\[ \delta(q_0, Y) = (q_3, Y, R) \quad \text{FOR A MATCHING 1 — BUT} \]
\[ \text{IF Y IS SEEN, GO FOR ENDGAME} \]

\[ \delta(q_1, 0) = (q_1, 0, R) \quad \text{SKIP OVER 0'S, Y'S TILL} \]
\[ \delta(q_1, Y) = (q_1, Y, R) \quad \text{1 IS FOUND — REPLACE} \]
\[ \delta(q_1, 1) = (q_4, Y, L) \quad \text{IT BY Y AND START} \]
\[ \text{HEADING BACK TO LEFT} \]

\[ \delta(q_2, Y) = (q_2, Y, L) \quad \text{MOVE LEFT SKIPPING 0/Y} \]
\[ \delta(q_2, 0) = (q_2, 0, L) \quad \text{TILL FIRST X IS FOUND} \]
\[ \delta(q_2, X) = (q_0, X, R) \quad \text{— MOVE RIGHT TO LOOK} \]
\[ \delta(q_3, Y) = (q_3, Y, R) \quad \text{FOR LEFTMOST 0} \]
\[ \delta(q_3, \eta) = (q_3, \eta, R, \eta) \quad \text{ENDGAME — MAKE SURE NO} \]
\[ \text{EXTRA 1'S LEFT OVER.} \]
Transition Diagram

Diagram showing states and transitions labeled with symbols like 0, 1, Y, and X.

Notation:
A/B → means replace A by B on tape cell being scanned, move right.

Remark:
Lots of transitions undefined — if input does not meet desired format, the T.M. will get "stuck" and halt in non-final state.

I.D.:
Used to show execution like in PDAs.

I.D. = q₁, q₁q₂ with qₖ ∈ Q
q₁, q₂ ∈ Σ*

Means:
Non-blank portion of tape has q₁q₂ with head at leftmost symbol.
Thus \( q \) \( d_1 q d_2 \) corresponds to

\[
\begin{array}{c}
\text{Blanks} \quad \alpha_1 \quad \alpha_2 \quad \text{Blanks} \\
\uparrow \\
\text{State } q.
\end{array}
\]

As before we use \( \vdash^* \) to show ID's changing.

**Example**

\[
q_0011 \vdash xq_1011 \vdash xoq_111 \\
\vdash xq_2oy_1 \vdash q_2xoy_1 \\
\vdash xq_1oy_1 \vdash xxq_1y_1 \\
\vdash xxq_1y_1 \vdash xxq_2yy \\
\vdash xq_2xyy \vdash xxq_0yy \\
\vdash xxq_3y \vdash xxyyq_3 \\
\vdash xxyybyq_4.
\]

**Language**

Given DTM \( M \)

\[
L(M) = \{ w \mid q_0w \vdash^* d_1p d_2 \} \\
\text{where } |EF \text{ and } \alpha_1, \alpha_2 \in \Gamma^* \}
\]

**Remark**

We use language recognition as a convenient notion of problem-solving ability.

However, T.M. can easily compute functions and produce output by leaving it on the tape.

Recursively Enumerable Languages

Class of Languages accepted by TM.
PROGRAMMING TRICKS.

**Idea** We present some notational conveniences which make it easier to "program" T.M. and also serve to highlight their generality and power.

**Basically** we impose notational structure on states and tape symbols.

**Trick 1** [CPU registers — using states as memory store]

Idea allow state names to be of the type 
\[ [q_1, x_1, \ldots, x_k] \]

where \( x_i \) acts as memorized symbols.

**Example** \( L = \{ w w^R \mid w \in \{0,1\}^* \} \)

**Define** \( M \)

\( Q = \{ [q_1, -], [q_0, 0], [q_1, 1], [s_1, 0], [s_1, 1] \} \)

\( \Sigma = \{ 0, 1 \} \quad \Pi = \{ 0, 1, \beta \} \)

\( q_0 = [q_1, -] \quad F = \{ \theta \} \)

**Idea** given \( w w^R \), match leftmost symbol with rightmost, erasing both.

**Use** CPU register to store leftmost symbol while heading right to find rightmost
**TRANSITIONS**

**STEP 1**
\[
\begin{align*}
S([q, -], 0) &= ([q, 0], B, R) \\
S([q, -], 1) &= ([q, 1], B, R) \\
S([q, -], B) &= (\emptyset, B, R)
\end{align*}
\]

*REMEMBER LEFTMOST SYMBOL & REPLACE BY BLANK, OR ACCEPT $\emptyset$*

**STEP 2**
\[
\begin{align*}
S([q, i], j) &= ([q, i], j, R) \\
\quad \forall i, j \in \{0, 1\}
\end{align*}
\]

*SKIP TO RIGHTMOST SYMBOL*

\[
S([q, i], B) = ([q, i], B, L)
\]

**STEP 3**
\[
\begin{align*}
S([s, i], i) &= (s, B, L) \\
\quad \forall i \in \{0, 1\}
\end{align*}
\]

*MATCH RIGHTMOST WITH REGISTER*

**STEP 4**
\[
\begin{align*}
S(s, i) &= (s, i, L), \forall i \\
S(s, B) &= ([q, -], B, R)
\end{align*}
\]

*SKIP OVER TO LEFTMOST SYMBOL*

---

**TRICK 2**

MULTIPLE TRACKS

IDEA: VIEW TAPE AS HAVING MULTIPLE TRACKS AND $\Gamma$ AS HAVING COMPOSITE SYMBOLS

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

THUS $\Gamma$ NOW CONTAINS SYMBOLS SUCH AS

\[
\begin{bmatrix}
0 \\
1 \\
X
\end{bmatrix}, \quad \begin{bmatrix}
Y \\
1 \\
0
\end{bmatrix}, \quad \begin{bmatrix}
0 \\
2 \\
1
\end{bmatrix}
\]

NOTE: UNNECESSARY
Consider \( L = \{ w w \mid w \in \{0,1\}^+ \} \)

Note first needs to find mid-point, and then we can use matching process as in \( w w^R \) to find mid-point we view tape as 2 tracks

\[
\begin{array}{c|c|c|c}
\text{0} & \text{1} & \text{1} & \text{0,1,1} \\
\end{array}
\]

Where we use the top track to put markers over symbols.

Idea: put markers on leftmost/rightmost symbols and slowly move them in till they meet at the mid-point.

Tape symbols: \([B], [B], [B], [0], [1], [0], [\ast], [\ast]\)

Could call these \( B, 0, 1, A, B \) but it would be less insightful in general.

Assume initially in state \( q_0 \), scanning leftmost

\[
S(q_0, [B]) = (q_1, [\ast], R)
\]

\[
S(q_1, [B]) = (q_1, [B], R)
\]

\[
S(q_1, [B]) = (q_1, [B], R)
\]
\[ \delta(q_0, [\mathbf{B}]) = (q_1, [\mathbf{B}], L) \]
\[ \delta(q_1, [\mathbf{B}]) = (q_2, [\mathbf{B}], L) \]
\[ \delta(q_2, [\mathbf{B}]) = (q_3, [\mathbf{B}], L) \]
\[ \delta(q_3, [\mathbf{B}]) = (q_0, [\mathbf{B}], R) \]

**Note:** One each of above transitions for \( i \in \{0, 1\} \)

At end we have head pointing to first symbol of second \( w \) with a * above it, in state \( q_0 \)

---

**Trick 3**  
**[Subroutines | Procedure Calls]**

**Example:** Shifting over

Given \( ID_1 = \alpha \ q_i \times \beta \) \( \forall \alpha, \beta \in \sum^* \) \( \square \in \sum \)

Want \( ID_2 = \alpha \square \ q_i \times \beta \) \( \forall \alpha, \beta \in \sum^* \) \( \square \in \sum \)

Subroutine can be used repeatedly to create space in middle of the tape

For example, useful for implementing counters as a part of complex processes

\( \#0\# \rightarrow \#1\# \rightarrow \#\square1\# \rightarrow \#01\# \)

\( \rightarrow \#10\# \rightarrow \#11\# \rightarrow \#\square11\# \)

\( \rightarrow \#011\# \rightarrow \#100\# \rightarrow \ldots \)
PROCEDURE CALL

\[ s(q_i, x) = ([P, x], [\hat{\delta}], R), \forall x \in \Gamma \]

- MEMORIZE RETURN STATE \( p_i \) ERASED SYMBOL \( x \)
- STATE \( p \) INVOKES PROCEDURE

PROCEDURE \( p \)

1) SHIFT 1 CELL TO THE RIGHT

\[ s([P, x], y) = ([P, y], x, R) \]
\[ \forall x, y \in \Gamma \text{ with } y \neq \text{B} \]

2) TILL REACHED END OF \( \text{B} \)

\[ s([P, y], \text{B}) = ([\epsilon, y, L]) \quad \forall y \in \Gamma \]

3) RETURN TO CALLING POINT

\[ s([\epsilon, y]) = ([\epsilon, y, L]) \quad \forall y \in [\hat{\delta}] \]

4) EXIT RETURN TO STATE \( p_i \)

\[ s([\epsilon, [\hat{\delta}]], \text{B}) = (q_i, [\square], R) \]

**NOTE** WE CAN IMPLEMENT ANY KIND OF PROCEDURE FUNCTION CALL WITH ANY KIND OF PARAMETER PASSING — WITH ENOUGH WORK.

**TEXTBOOK** SEE SIMPLER NOTION OF SUBROUTINES
ENHANCING T.M.

**Observe** if T.M. seen so far is able to capture all that we can compute, then adding features to it should not enhance its power.

**We show** adding following features gives T.M. which can be easily simulated by our standard T.M.

- **Multiple Tapes/Heads**
- **Non-determinism**

**Exercise** try to think of other ways to enhance T.M. and whether standard T.M. can simulate these features.

**Multi-tape T.M.**

![Diagram of multi-tape T.M.]

Initially, input w is on tape 1 with tape-head at leftmost symbol — remaining tapes are blank.

Each head independent & part of transition

\[ s(q_j, h_1, h_2, ..., h_k) = (p, (w_1, m_1), (w_2, m_2), ...) \]

- \( h_i \) symbol under head \( i \)
- \( w_i \) write in \( h_i \)
Consider simulating 2-tape T.M. \( M_R \) by 1-tape T.M. \( M_1 \)

**Idea**

Use 2\( k \) tracks in \( M_1 \) — for each tape of \( M_R \)
Use one track of \( M_1 \) to store tape contents
And another track to mark head position with *.

<table>
<thead>
<tr>
<th>TAPE 1</th>
<th>A</th>
<th>B</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>HEAD 1</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>TAPE 2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>HEAD 2</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TAPE 3</td>
<td>0</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>e</td>
</tr>
<tr>
<td>HEAD 3</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td></td>
</tr>
</tbody>
</table>

In \( M_1 \) each transition of \( M_R \) is simulated by a whole series of transitions

**Step 0**
Start at leftmost cell where some track contains a non-blank symbol

**Step 1**
Sweep right — picking up each \( h_i \) by noting symbol marked by each *; storing \( h_i \)'s in “CPU register”

**Step 2**
Sweep left — writing \( w_i \) and moving *’s as per \( M_R \)’s transition

**Verify**
Can construct \( M_1 \) to simulate \( M_R \) without affecting the language.

**Remark**
From now on, I will provide only such a high-level view of T.M. constructions — low-level programming details are very cumbersome and omitted — just verify T.M. can be constructed...
SIMULATION SPEED.

Observe while enhancements do not affect the power of T.M., they do impact its efficiency.

Running time A T.M. M is said to have running time $T(n)$ if it halts within $T(n)$ steps on all inputs of length $n$ (Note $T(n)$ could be infinite).

**THEM** If M(R) has running time $T(n)$, then M(R) will simulate it with running time $O(T(n)^2)$.

**Proof** Suppose input w has length n. Then M(R) can use $T(n)$ time on it.

Claim The R heads of M(R) cannot be more than $2T(n)$ apart and M(R) uses $\leq 2T(n)$ tape cells on each tape.

Why? At each step, leftmost and rightmost tape heads can drift apart by at most 2 additional cells.

Consider M(R)

- Makes 2 sweeps for transition of M(R)
- Time per sweep is $O(T(n))$
- Number of transitions of M(R) is $\leq 2T(n)$

Thus total time = $O(T(n)^2)$. 