In previous sections, we've learned regular languages and context free languages. Today, we'll learn about Turing Machines.

**Turing Machines (TM)**

A Turing machine is a theoretical model with a memory tape. It can read and write memory at any location.

![Diagram of a Turing Machine]

Finite Control = program (transition) assembly code

by providing a transition, in essence we are providing assembly code.
Memory Tape

- It is a one-directional tape, in the sense that it has a boundary on the left end, but without boundary on the right hand side.
- The tape is divided into cells in which a symbol is stored.
- The r/w head can read a symbol off the tape and can write a symbol into a cell. The r/w head can also move to the left and/or to the right.
- It can also be used as an input/output device.
  - At the beginning the input is given on the tape.

```
+----+----+----+----+----+
|    |    |    |    |    |
|<---|    |    |    |    |
|     |    |    |    |    |
|     |<---|    |    |    |
|     |    |    |    |    |
|     |    |    |    |    |
+----+----+----+----+----+
```

```
Input

|<---|    |    |    |    |
|    |    |    |    |    |
|    |    |    |    |    |
|    |    |    |    |    |
```

```
After 2 moves:

|<---|    |    |    |    |
|    |    |    |    |    |
|    |    |    |    |    |
|    |    |    |    |    |
```

```
finite control
```

```
q
```
Turing Machines: Formal Definition

A Deterministic Turing Machine (DTM) $M$ is a 5 tuple:

$$(S, \Sigma, \delta, s, H)$$

$S$ is a finite set of states

$\Sigma$ is an alphabet of state symbols containing a LEFT end marker: $\rhd$

and a blank symbol: $\_\_$ in addition to symbols required to describe the underlying problem, either in function or languages.

when $\rhd$ is read, you cannot go to the left.
when $\_\_$ is seen, you just read a string & that’s your input.

$s$ is the start/initial state

$H$ is a subset of $S$; it is a set of halting states.

$\delta$ is a transition function (finite control, program)

$\delta: (S-H) \times \Sigma \rightarrow S \times (\Sigma \cup \{ \leftarrow, \rightarrow \})$

Example:

After 2 moves:
$\delta(p,0) = (q,1)$ and $\delta(q,1)=(q,\rightarrow)$

The TM does the following:
At state $p$, if the head points to a 0, overwrite 0 with 1, change to state $q$, and then move to the right.

A TM can perform the following basic operations:
1. It can overwrite a symbol without moving the head.
2. It can move the head, either to the left or the right, without overwriting.
3. It can simply stay on the same place but change its state.
Example:

\[ \delta(p,a) = (p,b) \]
- overwrite, no change of state

\[ \delta(p,a) = (q,b) \]
- overwrite, change its state

\[ \delta(p,a) = (q,\rightarrow) \]
\[ (q,\leftarrow) \]
- change location

\[ \delta(p,a) = (q,a) \]
- just change its state

Restrictions for Transitions

1. If current symbol pointed to is \( \rightarrow \), then the head is not allowed to move to the left; namely we cannot have:

\[ \delta(p, \rightarrow) = (q,\leftarrow) \]

2. As soon as a halting state is reached, the machine stops.

See example 4.1.1 on page 124

A TM can be described at three levels:

1. Low level: formal definition with transition function
2. Middle level: between low level and high level (often use modules and diagram)
3. High level: English + mathematical notations

Middle Level Examples:
M;
if a then A else B;

while (not a) do A;

a:=b;
if (a \neq \text{ | |}) then move Right, then write b

while (not \text{ | |}) do \text{ | |} R
(while not blank, write \text{ | |} and move to the right)

while (not \text{ | |}) do R

Read Examples 4.1.3