Defn: A (one-tape) Turing Machine is a 7-tuple $M = (Q, \Sigma, \Gamma, s, \omega, s, F)$ where:

- $Q$ is a set of states like with a FA.
- $\Sigma$ is the input alphabet (or I/O alphabet if we allow functional output).
- $\Gamma$, a superset of $\Sigma$, is the work alphabet.
- $\omega$, also written 'B' or just a space, is the blank char, which is in $\Gamma \setminus \Sigma$.
- $s \in Q$ is the start state (aka $q_0$).
- $F \subseteq Q$ is the set of finals states (aka $F = \{q_{acc}\}$ in text).
- $S$ is a subset of $Q \times \Gamma \times (\Gamma \times \{L, R, S\} \times \Gamma \times S)$ (aka tuple).

Typical instruction: $[p, c, d, D, g]$ $[c \bar{d}, D]$, $(d = c$ allowed $)$ $[c \bar{d}, D]$, $[c, \bar{d}, D]$.

Typically, $M$ can read $c$, change it to $d$, move in direction $D$ (with $S = S_{Init}$) and go to state $q$.

$M$ is deterministic if no two instructions have the same $(p, c, \_\_\_\_)$ beginning.

Then we can say $S$ is a (partial!) function from $Q \times \Gamma \rightarrow \Gamma \times \{L, R, S\} \times \Gamma \times Q$.

In the texts normal form, a OTM has $S: (Q \times \{Eq, acc, rej\} \times \Gamma) \rightarrow \Gamma \times \{L, R, S\} \times (Q \times \{Eq, acc, rej\} \times \Gamma)$.

[Turing Kit Demo] Was Long

Defn: An (instantaneous description) of a Turing Machine is a triple $I$ that specifies:

- the current state $q$
- the current symbol $w$
- the current position $h$ of the tape head, with $h \geq 0$ when it started

$I = (q, w, h)$

For a one-way-tape TM, cell 0 is the left end.

Turing Kit shows a two-way infinite tape.
Definition of Computation: Given TMs I, J of a system, write $I \vdash_m J$ if there is an instruction in the TM $I$ that when executed in $I$ gives the TM $J$.

In configuration $I, M$ can go to $J$ in one step. (This notation applies to multiple-tape Turing machines as well.)

Then we write:

$I \vdash^0 M I$ in 0 steps.

$I \vdash^k M J$ if there is an $I' I$, st. $I \vdash^k M I'$ and $I' \vdash^k M J$.

$I \vdash^* M J$ if for some $K \geq 0$, $I \vdash^k M J$.

An TM is accepting if its state $q$ belongs to $F$ and there is no instruction defined for $q$ and $c$ where $c$ is in position $h$ of $w$.

$\text{Def. of Computation: Given TMs I, J of a system, write } I \vdash_m J$ if there is an instruction in the TM $I$ that when executed in $I$ gives the TM $J$.

In configuration $I, M$ can go to $J$ in one step. (This notation applies to multiple-tape Turing machines as well.)

Then we write:

$I \vdash^0 M I$ in 0 steps.

$I \vdash^k M J$ if there is an $I' I$, st. $I \vdash^k M I'$ and $I' \vdash^k M J$.

$I \vdash^* M J$ if for some $K \geq 0$, $I \vdash^k M J$.

An TM is accepting if its state $q$ belongs to $F$ and there is no instruction defined for $q$ and $c$ where $c$ is in position $h$ of $w$.

$\text{Def. of Computation: Given TMs I, J of a system, write } I \vdash_m J$ if there is an instruction in the TM $I$ that when executed in $I$ gives the TM $J$.

In configuration $I, M$ can go to $J$ in one step. (This notation applies to multiple-tape Turing machines as well.)

Then we write:

$I \vdash^0 M I$ in 0 steps.

$I \vdash^k M J$ if there is an $I' I$, st. $I \vdash^k M I'$ and $I' \vdash^k M J$.

$I \vdash^* M J$ if for some $K \geq 0$, $I \vdash^k M J$.

An TM is accepting if its state $q$ belongs to $F$ and there is no instruction defined for $q$ and $c$ where $c$ is in position $h$ of $w$.

$L(M) = \{ x \in \Sigma^* : M \text{ runs from } I_0(x) \text{ enters an accepting TM} \}$

$I_0(x) = q = 5, w = x, h = 0$. (can force $J = \langle q_{acc}, 1, 0 \rangle$.