A Turing Machine (TM) allows to decide if a TM, DTMs liberalizes a DFA (or NFA) by:

- allowing to change states on one or more tapes
- allowing tape heads to move left (L) or stay stationary (S) beside moving right (R).

Upshot: TMs can decide languages like \[ \{ a^n b^n c^n : n \geq 0 \} \]
that are not even CFLs, let alone regular.

The work alphabet \( \Gamma \) always includes \( \Sigma \) plus
- blank \( B \), cannot include other states
- \( \Lambda, B, \#, \times \)

The initial code of my TM can emulate a DFA
Mo such that
\[ L(M_0) = a^+ b^+ c^+ \]
(not necessarily)

So

\[ (a/a, R) \]
\[ (b/b, R) \]
\[ (c/c, R) \]

Rewind to \( \Lambda \)

\[ (\delta(\delta(a, B), R), L) \]
\[ (\delta(\delta(b, B), R), L) \]
\[ (\delta(\delta(c, B), R), L) \]

\[ d = a, b, c \text{ or } \times \]
**Defn:** A Turing Machine is a 7-tupel $M = (Q, \Sigma, \Gamma, \delta, \in, q_0, F)$, where:

- $Q$ is a finite set of states
- $\Sigma$ is the finite input alphabet
- $\Gamma$ is the blank (\_ in text, or \texttt{0}, or \texttt{1} etc.) tape alphabet
- $\delta$, which always includes $\Sigma \cup (\Gamma \setminus \Sigma)$, is the transition function
- $\in$, which always includes $\Sigma \cup (\Gamma \setminus \Sigma)$, is the work alphabet
- $q_0$ is the start state (\texttt{0} \ in \ text)
- $F$ is the set of desired final states

$\delta \subseteq Q \times \Gamma \times \Gamma \times \{L, R, S\} \times Q$

Typical cell or instruction tuple $(p, c, d, D, q)$

Diagram

Furthermore:

- $M$ is **deterministic** if for all $p \in Q$ and $c \in \Gamma$, there is at most one tuple in $\delta$ that begins $(p, c/---)$.
- $M$ is **complete** if for all $p \in \{q_{acc}, q_{rej}\}$ and $c \in \Gamma$, there is a tuple beginning $(p, c/---)$.

Together $\Rightarrow \delta$ is a function from $(Q \setminus \{q_{acc}, q_{rej}\} \times \Gamma)$ to $(\Gamma \times \{L, R, S\} \times Q)$, a text defn of a DTM.

Otherwise, if there is any pair $(q, c)$ with two or more rules beginning $(q, c/---)$ then $M$ is properly an NTM.
Extension for any number \( k \) of tapes. \( M = (Q, \Sigma, \Gamma, \delta, \delta_0, \delta_f) \)

with \( s = Q \times \prod_{i=1}^{k} \Gamma_i \times \prod_{i=1}^{k} \Gamma_i \times \{L, R, S\}^k \times Q \)

Start

\( k = 2 \)

Initially

\( X_a X_a X_a X_a X_a X_b | B | \quad B - B - B - B - B \)

Initially

\[ (a \downarrow a, S) \quad (a \downarrow a, R) \quad (b \downarrow b, R) \quad (b \downarrow b, L) \]

L(M) = \{a^m b^n a^m b^n : m, n \geq 0 \mid (\$1-\)

which is a CFL.

A DFA

NFA \( 2 \rightarrow A \) 1-tape TM in which every tuple \((p, c/d, D, g)\)

has \( d = c \) and \( D = R \).

\( \Rightarrow \) A

DPDA

N8DA \( 2 \rightarrow A \) 2-tape TM in which every instruction

\[ (P_1, C_1, d_1, D_1, q) \]

has \( d_1 = C_1 \) and \( D_1 \neq L \)

Type 2

is a

pushdown

Automaton.

Added: The definition of deterministic/non-deterministic is similar for all these forms.