- what if a TM has more tapes? several heads?

This section: we’ll give detailed descriptions of our machines but not give detailed $\delta$-functions.
Multitape Turing Machines

- have to redefine $\delta$-function:

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

2nd tape
1st tape

Thm 3.13: Every multitape TM has an equiv. single-tape TM.

Note:
- TM accepts if $q = q_{\text{accept}}$
- if 1st tape wants to rewrite $0$, then need to shift the 2nd tape to the right

Standard TM: place tapes after each other

New states:
- contain, multitape state $(q)$
- symbol read on the 1st tape $(a)$
- symbol on the 2nd tape $(b)$

Symbol for the tape separator

1) scan to find the 2 dotted symbols $a, b$
2) state knows $q$, then can simulate $\delta(q,a,b)$
3) scan to replace $a \rightarrow c_1$, $b \rightarrow d$ and update dotted symbols based on the direction $L,R,S$
4) update state info $(q)$
Nondeterministic Turing Machines

- have to redefine $\delta$-function:

$$\delta : Q \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{L, R, s\})$$

Thm 3.16: Every nondeterministic TM has an equivalent deterministic TM.

**idea:**

configuration computation tree for NTM:

DTM:

1) shift the input $w$ to the right and precede it by $q_0$, after input $#$
2) underline the configuration (current config)
3) for the current config,
   - place all subsequent config. at the end of the tape, delimited by $#$
4) back to the current (underlined config.)
5) ununderline the config & underline the next one
6) repeat 3) until find $q_{accept}$
An alternative name for Turing-recognizable languages is **recursively enumerable** languages.

An enumerator is a TM-like “printer” with no input and an extra output tape that prints all strings in a given language.

For a TM, let’s create an enumerator printing the same lang.

**Idea 1:**
1) Start w. $\varepsilon$, simulate TM, if accepts, print $\varepsilon$, if rejects move to 2)
2) Generate the next string in the lexicographic ordering, run 1) on that string

**Problem:** what if the TM does not halt on a string (but there are strings accepted later in the lexic. ordering)

**Idea 2:**
- $\text{go } \#a \#b \#a a$
- 1 step
- 1 step
- 1 step 1 step 1 step

Next: add a new string, perform 1 step of the TM on all strings

$\Rightarrow$ if accept, print the string.
Thm 3.21: A language is Turing-recognizable iff there is an enumerator for it.