Recap

- TM definition
- Doubly-infinite tape
- Multi-tape
- Non-deterministic TM
- Church-Turing thesis

Doubly Infinite tape vs. canonical TM

- Claim: they are equivalent.

Canonical:                          Doubly-infinite

\[ \text{control} \rightarrow \text{infinite} \rightarrow \cdots \]

\[ \cdots \rightarrow \text{control} \rightarrow \text{infinite} \rightarrow \cdots \]

Provide a compiler from one model to the other, and vice versa. Prove the correctness of each of these compilers.

First we simulate a canonical TM using a doubly infinite TM.

- At the beginning, move left, put a # (special symbol not in input) in that cell, move right.
- At any point in computation, if the TM sees a #, move right and stay in the same state.

Other direction:

- At the beginning, mark the left most cell with a mark: a \( \rightarrow \) a. Skip to the right most cell with input and mark it as well: a \( \rightarrow \) a. Return to the left most cell.
- If at any point in computation, the TM wants to go left from a cell marked a, enter a subroutine that shifts all input between a and a to the right one cell. Finally erase the top mark over what was the first symbol.
and put it over the \( \sqcup \) at the beginning of the tape.

- if at any point in computation the TM wants to move right from a cell marked with the bottom mark, erase the mark and place it under the blank cell to the right.
- if ever overwriting a cell with a mark, do not erase the mark. only rewrite the symbol.

**Multi-tape TM**

**vs. Canonical TM**

- **Multi-tape TM**: has \( K \) tapes and \( K \) heads.

\[ \begin{array}{cccc}
\sqcup & b_1 & b_2 & \ldots & b_k & \sqcup & \ldots \\
& & & K
\end{array} \]

- Each transition of the form:
  \[ a_1 \ldots a_K \rightarrow b_1 \ldots b_k \quad d_1 \ldots d_K \]
  \[ d \in \{ L, R \} \]
  \[ a, b, d \in \Gamma, \quad 1 \leq d \leq K \]

- To simulate a canonical TM on a multi-tape TM, ignore all tapes except the first one.
To simulate a multi-tape TM using a canonical TM, do the following:

First, we make an encoding scheme for K tapes.

The dot above a cell value $\&$ means has a head pointer pointing to it.

So on input $w_1 \ldots w_n$, first put the tape in the correct format:

\[ \# w_1 \# w_2 \# \ldots \# w_n \# \ldots \# \# \]

To simulate a transition of the K-tape machine:

- Scan tape from $\#_0$ to $\#_k$ to identify each input for the K-tape transition function.
- Execute the transition, write new symbols under the correct heads and move the heads accordingly.
- If needed, make more room for any of the K tape locations by shifting the rest of the tape's contents to the right.
- If the machine we are simulating accepts, do the same.
Nondeterministic TM: transition function may have >1 transition for some situations.

\[ S: \{ a, b, \text{ halts} \} \times \Gamma \rightarrow P(\{ a, b, \text{ halts} \}) \]

- If \( N \) is an NTM, its configuration is encoded by \( x_1, x_2 \). \( x_1 \) is the contents of the tape to the left of the head, \( x_2 \) is the cell of the head, and \( x_2 \) is the remainder of meaningful input.

\[ C_1 = x_1, x_2 \text{ yields } C_2 = x_1, x_2' \text{ if some transition from } C_1 \text{ shows it.} \]

- \( C_0, C_1, \ldots, C_t \) is a valid computation history of \( N \) on input \( w \) for time \( t \) if:
  1. \( C_0 = \text{start } w \)
  2. \( C_i \text{ yields } C_{i+1} \text{ for } 0 \leq i < t \)

- \( N \) accepts \( w \) if \( \exists \) \( t \) and \( C_0, \ldots, C_t \) s.t.
  it is a valid computation history of \( N \) on \( w \) and \( C_t \) has accept.

\[ L(N) = \{ w \mid N \text{ accepts } w \} \]

**Theorem:** If \( L = L(N) \) for an NTM \( N \), then \( \exists \) TM \( M \) s.t. \( L = L(M) \). We will show by simulating an NTM on a 3-tape TM.

```
\begin{array}{c}
\text{Control} \\
\hline
\text{Input tape} \\
\text{(read only)} \\
\end{array}
```

```
\begin{array}{c}
\text{Work tape} \\
\text{List of choices; brute force by trying every } \{ C_1, C_2, \ldots, C_t \} \text{ non-det. tape.} \\
\text{Possible list of choices for } 1 \leq C_i \leq t \\
\end{array}
```

```
\begin{array}{c}
\text{List of choices; brute force by trying every } \\
\end{array}
```