An Aside on Algorithms

• “An algorithm is a procedure with instructions so detailed that no further information is necessary”

• **Goal:** Create a “universal algorithm machine”

• In 1936, Emil Post created a Post machine – which he hoped would be a “universal algorithm machine”

• **Requires:** Universal algorithm machines must accept *any language* which can be defined by humans
A Post Machine, denoted PM, is a collection of five things:

1. An alphabet $\Sigma$ of input letters and the special symbol $\#$
2. A linear storage location called the STORE. We can read the leftmost character in the store and add a new character to the “end” (rightmost location) of the STORE. We allow for characters not in $\Sigma$ to be used in the STORE — usually denoted as $\Gamma$.
3. READ states which remove the leftmost character from the STORE and branch accordingly.
ADD states which concatenate a character onto the right end of the string in the STORE. (This is the “opposite” of a PDA PUSH state). No branching can take place. Letters from $\Sigma$ and $\Gamma$ can be Added to the STORE.

A START state (unenterable) and some halt states called ACCEPT and REJECT. REJECT states are optional.
Example

Trace: aaabbb
Example #2

START

ADD #

READ₁

READ₂

READ₃

READ₄

ACCEPT

ADD a

ADD b

ADD a

ACCEPT

ADD a
Simulating a PM on a TM

Theorem

Any language that can be accepted by a PM can be accepted by some TM

Proof.

• START states remain unchanged
• ACCEPT states can be renamed to HALT
• REJECT states can be removed
• READ states should move the TAPE-HEAD to the first non-$\Delta$ character on the TAPE.

$\begin{align*}
(a, a, L) \\
(#) , (#, L) \\
(b, b, L)
\end{align*}$
Simulating a PM on a TM

Proof.

- ADD states should move the TAPE-HEAD to the “end” of the tape and insert the character to the END

\[(a, a, R)\]
\[(b, b, R)\]
\[(#, #, R)\]

\[(\Delta, y, L)\]
Simulating a TM on a PM

**Theorem**

*Any language that can be accepted by a TM can be accepted by some PM*

**Proof.**

- Key: use # to indicate the “tape” boundary separator
- TAPE may store any of $\Sigma$, $\Gamma$, #, $\Delta$
- TAPE-HEAD will always be the front of the STORE
- When we *read* from the TM, we READ from the PM
- When we *write* to the TM, we ADD to the PM
- When we move to the *left*, we have to rotate all of the elements in our STORE right (cyclically)
- When we move to the *right*, we don’t have to do anything
- START needs a secondary *ADD #* state immediately after. Any cycles will go to this new ADD state
Simulating a TM on a PM

Converting transition of $(X, Y, R)$

Converting transition of $(X, Y, L)$

Changing START
$TM = PM$

Proof.

- $PM \subseteq TM$ because we can show how to convert a PM to a TM
- $TM \subseteq PM$ because we can show how to convert a TM to a PM
- $PM = TM$ because of the above two claims