Next Tuesday quiz on definitions.

Turing machine & μ-recursion
Ø, 0, 1, 10, 0, 1, 12, μ-recursion

0 (a, a2)

start, # "in lst, L
in lst, 1 → R
in lst, # → R, in 2nd
in 2nd, 1 → R
in 2nd, # → er 2nd, L
er 2nd, 1 → #, L
er 2nd, # → er lst, L
er lst, 1 → L, #
er lst, # → halt

---

[Diagram of a Turing machine state transition graph]
How to describe composition?

We have TM for computing $P(n), g(n)$
we want: a TM for computing $h(g(n))$

$g(n) = n+1$

<table>
<thead>
<tr>
<th>start</th>
<th>$# \rightarrow$ working, R</th>
</tr>
</thead>
<tbody>
<tr>
<td>working</td>
<td>$# \rightarrow L$, halt</td>
</tr>
<tr>
<td>testing</td>
<td>$L \rightarrow R$, working</td>
</tr>
<tr>
<td>working</td>
<td>$L \rightarrow R$</td>
</tr>
<tr>
<td>ready</td>
<td>$# \rightarrow$ back, L</td>
</tr>
<tr>
<td>back</td>
<td>$L \rightarrow L$</td>
</tr>
<tr>
<td>back, $# \rightarrow$ halt</td>
<td></td>
</tr>
</tbody>
</table>

Subtracting 1 in unary code:

```
start, $\# \rightarrow$ testing, R

$P(n) = n-1$
```

Ex.

```
  $\# \# \# \# \# \#$

   $\uparrow$
start

  $\# \# \# \# \# \#$

   $\uparrow$
testing

  $\# \# \# \# \# \#$

   $\uparrow$
working

  $\# \# \# \# \# \#$

   $\uparrow$
working

  $\# \# \# \# \# \#$

   $\uparrow$
working

  $\# \# \# \# \# \#$

   $\uparrow$
working

  $\# \# \# \# \# \#$

   $\uparrow$
working
```

```
  $\# \# \# \# \# \#$

   $\uparrow$
ready

  $\# \# \# \# \# \#$

   $\uparrow$
back

  $\# \# \# \# \# \#$

   $\uparrow$
back

  $\# \# \# \# \# \#$

   $\uparrow$
halt
```
General algorithm: \( f(g(n)) \)
- Rename states so that they are different
- Instead of halt, you place start

Working from TM1 becomes working
Working from TM2 becomes working

TM1 = \( g(n) = n+1 \)
TM2 = \( f(n) = n-1 \)

\[ f(g(n)) : \]

```
+----+
|     |
| 1   |
|     |
```

start
```
+----+
|     |
| 1   |
|     |
```
working_1
```
+----+
|     |
| 1   |
|     |
```
working_2
```
+----+
|     |
| 1   |
|     |
```
back_1
```
+----+
|     |
| 1   |
|     |
```
back_2
```
+----+
|     |
| 1   |
|     |
```
back_2, halt
```
+----+
|     |
| 1   |
|     |
```
```
+----+
|     |
| 1   |
|     |
```
```
Projection:

9.2. We have (2.1)

\[ \begin{align*}
\text{start} & \rightarrow \text{inlst}, R \\
\text{inlst}, 1 & \rightarrow R \\
\text{inlst}, \# & \rightarrow \text{in2nd}, R \\
\text{in2nd}, 1 & \rightarrow R \\
\text{in2nd}, \# & \rightarrow \text{erase}, L \\
\text{erase}, 1 & \rightarrow \#, L \\
\text{erase}, \# & \rightarrow \text{back}, L \\
\text{back}, 1 & \rightarrow L \\
\text{back}, \# & \rightarrow \text{halt}
\end{align*} \]
\[ \pi_2 \]

\[ (2,1) \text{ we have} \]

\[ \text{we want} \]

Mark end of tape so we don't fall off the cliff

Start, \# \rightarrow 1, erase, left, R

erase, left, 1 \rightarrow \#, R

erase, left, \# \rightarrow right, R

right, 1 \rightarrow R

right, \# \rightarrow erase, L

erase, L \rightarrow \#, L, left

left, L \rightarrow L

left, \# \rightarrow 1, B, right

left, \# \rightarrow 1, checking, L

checking, \# \rightarrow right, R

checking, 1 \rightarrow \#, halt
(0, 0) 
# #
^ start

# #
^ erase

# #
^ right

# #
^ erase

Add rule
erase, # \rightarrow \text{finish, } L
finish, # \rightarrow L
finish, 1 \rightarrow #, \text{halt}

Deterministic, for every state we have a rule for 1 and #.