Lecture 18. Universal Turing Machines

- So far, each Turing machine appears to be specialized at solving one particular problem.
- Can we 'program' a Turing machine to obtain a general-purpose programmable Turing machine?
  Yes, such a machine is called a Universal Turing machine; they can be *programmed* to solve any problem that can be solved by specialized Turing machines.
- In fact, the universal Turing machine played an important early role in stimulating the development of stored-program computers.

Universal Turing Machine, U

Input to U:

> The encoding of a TM M, denoted as "M", and
> the encoding of a string $w \in \Sigma^*$, denoted as "w".

Behavior:

> U halts on input "M" "w" iff M halts on w.
> (U emulates the running of M on w).

Encoding Turing machines and their input strings

How do we encode a Turing machine as a string?

Example:

$M = (K, \Sigma, \delta, s, H)$ where
$K = \{s, q, h\}$
$\Sigma = \{\sqcup, \triangleright, a\}$
$H = \{h\}$
δ :

State	Symbol	δ
s	a	$(q, \sqcup)$
s	$\sqcup$	$(h, \sqcup)$
s	$\triangleright$	$(s, \rightarrow)$
q	a	(s, a)
q	$\sqcup$	$(s, \rightarrow)$
q	$\triangleright$	$(q, \rightarrow)$

Encode the states:

$$\begin{array}{ccc} s & — & q00 \\ q & — & q01 \\ h & — & q10 \end{array}$$

The start state is always encoded as $q00...0$.

How to distinguish the halt states from other states? Since the transition function δ is defined only on $(K - H) \times \Sigma$, any state for which δ is not defined is a halting state. If $H = \{y, n\}$, we adopt the convention that y is the lexicographically smallest state and n is the second smallest.

Encode the symbols: $\Sigma \cup \{\rightarrow, \leftarrow\}$

$$\begin{array}{ccc}
\sqcup & \text{—} & a000 \\
\triangleright & \text{—} & a001 \\
\leftarrow & \text{—} & a010 \\
\rightarrow & \text{—} & a011 \\
a & \text{—} & a100
\end{array}$$

Encode the transitions:

Encode each transition as a quadruple.

$$\begin{array}{rcl}
\delta(s, a) = (q, \sqcup) & \text{—} & (q00, a100, q01, a000) \\
\delta(s, \sqcup) = (h, \sqcup) & \text{—} & (q00, a000, q10, a000) \\
\delta(s, \triangleright) = (s, \rightarrow) & \text{—} & (q00, a001, q00, a011) \\
\delta(q, a) = (s, a) & \text{—} & (q01, a100, q00, a100) \\
\delta(q, \sqcup) = (s, \rightarrow) & \text{—} & (q01, a000, q00, a011) \\
\delta(q, \triangleright) = (q, \rightarrow) & \text{—} & (q01, a001, q01, a011)
\end{array}$$

Encode a TM M and a string w:

$$\begin{aligned}\text{“}M\text{”} = &(q00, a100, q01, a000)(q00, a000, q10, a000)\\ &(q00, a001, q00, a011)(q01, a100, q00, a100)\\ &(q01, a000, q00, a011)(q01, a001, q01, a011)\end{aligned}$$

Note: It is sufficient to encode only the transition function, rather than also lists the encodings of the states and the input alphabet. We will show that such an encoding of a TM is sufficient for a UTM to simulate the behavior of the TM (on an input string w).

“w” = “$aa \sqcup a$” $= a001a100a100a000a100$
That is, a Turing machine is encoded as a string over the alphabet $\{q, a, 0, 1, (,), ,\}$, and a input of a TM is encoded as a string over the alphabet $\{a, 0, 1\}$.

The input to a UTM is “M” “w”.
For example,

$$\begin{aligned}\text{“}M\text{” “}w\text{”} = &(q00, a100, q01, a000)(q00, a000, q10, a000)\\ &(q00, a001, q00, a011)(q01, a100, q00, a100)\\ &(q01, a000, q00, a011)(q01, a001, q01, a011)\\ &a100a100a000a100\end{aligned}$$

In general:
Let $M = (K, \Sigma, \delta, s, H)$ be a TM.

1. Choose i such that $2^i \geq |K|$.
2. Choose j such that $2^j \geq |\Sigma| + 2$.
3. Encode each state in K as: q followed by a binary string of length i.
$$s \text{ — } q0^i$$
4. Encode each symbol in $\Sigma \cup \{\leftarrow, \rightarrow\}$ as: a followed by a binary string of length j.

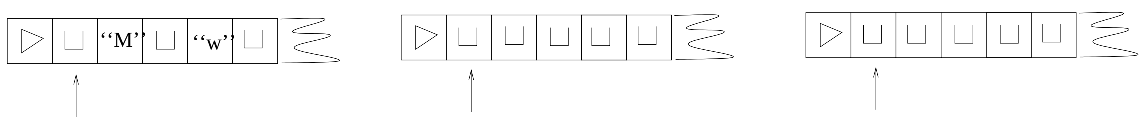
$$\begin{array}{ccc} \sqcup & \text{—} & a0^j \\ \triangleright & \text{—} & a0^{j-1}1 \\ \leftarrow & \text{—} & a0^{j-2}10 \\ \rightarrow & \text{—} & a0^{j-2}11 \\ \vdots & & \end{array}$$

Universal Turing Machine

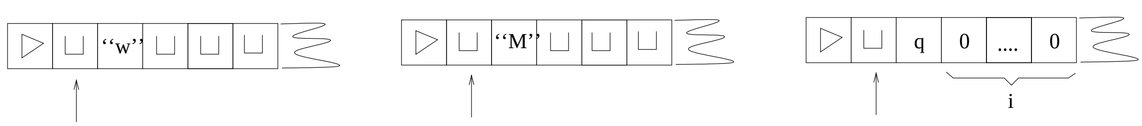
Instead of describing a UTM that has only one tape, we describe a UTM U that has 3 tapes.

U halts on input "M" "w" iff M halts on w.

1. Initially, Tape 1 contains "M" "w". Tape 2 and 3 are blank.



2. Move "M" onto Tape 2, and shift "w" down to the left end of Tape 1. Write the encoding of the initial state of M on Tape 3.



3. Scan Tape 2 to search for a quadruple (x, y, z, t) where x matches the current encoded state on Tape 3 and y matches the current encoded symbol scanned from Tape 1. If found,

   - write z, the encoding of the next state, to Tape 3,
   - perform the action (a write or a movement of head) indicated by t on Tape 1.

   If not found, the current state must be a halt state, so U halts.