Turing Machines

Sections 17.6 – 17.7
The Universal Turing Machine

**Problem:** All our machines so far are hardwired.

ENIAC - 1945
The Universal Turing Machine

**Problem**: All our machines so far are hardwired.

**Question**: Can we build a programmable TM that accepts as input:

\[\text{program} \quad \text{input string}\]

executes the program, and outputs:

\[\text{output string}\]
The Universal Turing Machine

Yes, it’s called the *Universal Turing Machine*.

To define the Universal Turing Machine $U$ we need to:

1. Define an encoding operation for TMs.
2. Describe the operation of $U$ given input $<M, w>$, the encoding of:
   - a TM $M$, and
   - an input string $w$. 
Encoding a Turing Machine $M$

We need to describe $M = (K, \Sigma, \Gamma, \delta, s, H)$ as a string:

- The states
- The tape alphabet
- The transitions
Encoding the States

• Let $i$ be $\lceil \log_2(|K|) \rceil$.

• Number the states from 0 to $|K|-1$ in binary:
  · Number $s$, the start state, 0.
  · Number the others in any order.

• If $t'$ is the binary number assigned to state $t$, then:
  · If $t$ is the halting state $y$, assign it the string $yt'$.
  · If $t$ is the halting state $n$, assign it the string $nt'$.
  · If $t$ is any other state, assign it the string $qt'$.
Example of Encoding the States

Suppose $M$ has 9 states.

\[ i = 4 \]

\[ s = q_{0000}, \]

Remaining states (where $y$ is 3 and $n$ is 4):

\[ q_{0001}, q_{0010}, y_{0011}, n_{0100}, \]
\[ q_{0101}, q_{0110}, q_{0111}, q_{1000} \]
The tape alphabet:

\[ ay : y \in \{0, 1\}^+, \]

\[ |y| = j, \text{ and} \]

\[ j \text{ is the smallest integer such that } 2^j \geq |\Gamma|. \]

Example: \( \Sigma = \{\square, a, b, c\} \). \( j = 2. \)

\[ \square = a00 \]
\[ a = a01 \]
\[ b = a10 \]
\[ c = a11 \]
Encoding a Turing Machine $M$, Continued

The transitions: \((\text{state}, \text{input}, \text{state}, \text{output}, \text{move})\)

Example: \((q000, a000, q110, a000, \rightarrow)\)

Specify $s$ as $q000$.

Specify $H$. 
A Special Case

We will treat this as a special case:

$q_0$
An Encoding Example

Consider $M = \{s, q, h\}, \{a, b, c\}, \{\square, a, b, c\}, \delta, s, \{h\}$:

$<M> = (q00, a00, q01, a00, \rightarrow), (q00, a01, q00, a10, \rightarrow), (q00, a10, q01, a01, \leftarrow), (q00, a11, q01, a10, \leftarrow), (q01, a00, q00, a01, \rightarrow), (q01, a01, q01, a10, \rightarrow), (q01, a10, q01, a11, \leftarrow), (q01, a11, h11, a01, \leftarrow)$
Enumerating Turing Machines

*Theorem:* There exists an infinite lexicographic enumeration of:

(a) All syntactically valid TMs.

(b) All syntactically valid TMs with specific input alphabet $\Sigma$.

(c) All syntactically valid TMs with specific input alphabet $\Sigma$ and specific tape alphabet $\Gamma$. 
Enumerating Turing Machines

Proof: Fix \( \Sigma = \{ (, ), a, q, y, n, 0, 1, \text{comma}, \rightarrow, \leftarrow \} \), ordered as listed. Then:

1. Lexicographically enumerate the strings in \( \Sigma^* \).
2. As each string \( s \) is generated, check to see whether it is a syntactically valid Turing machine description. If it is, output it.

To restrict the enumeration to symbols in sets \( \Sigma \) and \( \Gamma \), check, in step 2, that only alphabets of the appropriate sizes are allowed.

We can now talk about the \( i^{\text{th}} \) Turing machine.
Another Win of Encoding

One big win of defining a way to encode any Turing machine $M$:

- We can talk about operations on programs (TMs).
Example of a Transforming TM $T$:

**Input:** a TM $M_1$ that reads its input tape and performs some operation $P$ on it.

**Output:** a TMM$_2$ that performs $P$ on an empty input tape.
Encoding Multiple Inputs

Let:

\(<x_1, x_2, \ldots x_n>\)

mean a single string that encodes the sequence of individual values:

\(x_1, x_2, \ldots x_n\).
The Specification of the Universal TM

On input \(<M, w>\), \(U\) must:

- Halt iff \(M\) halts on \(w\).

- If \(M\) is a deciding or semideciding machine, then:
  - If \(M\) accepts, accept.
  - If \(M\) rejects, reject.

- If \(M\) computes a function, then \(U(<M, w>)\) must equal \(M(w)\).
How $U$ Works

$U$ will use 3 tapes:

- Tape 1: $M'$'s tape.
- Tape 2: $<M>$, the “program” that $U$ is running.
- Tape 3: $M'$'s state.
The Universal TM

Initialization of $U$:
1. Copy $<M>$ onto tape 2.
2. Look at $<M>$, figure out what $i$ is, and write the encoding of state $s$ on tape 3.

After initialization:
The Operation of $U$

Simulate the steps of $M$:
1. Until $M$ would halt do:
   1.1 Scan tape 2 for a quintuple that matches the current state, input pair.
   1.2 Perform the associated action, by changing tapes 1 and 3. If necessary, extend the tape.
   1.3 If no matching quintuple found, halt. Else loop.
2. Report the same result $M$ would report.

How long does $U$ take?
If A Universal Machine is Such a Good Idea …

Could we define a Universal Finite State Machine?

Such a FSM would accept the language:

\[ L = \{ <F, w> : F \text{ is a FSM, and } w \in L(F) \} \]