Encoding TMs and the Universal TM
It is all Zeros and Ones

One of the most popular and overemphasized clichés about computer professionals. Proxy for: genius / nerd / insensitive / anti-social / ...
(still bits are quite fundamental, “more than Atoms”)
Bit Strings Encoding

Encode a finite string in $\Sigma^*$ as a bit string: encode each character as $\log |\Sigma|$ bits.

For $x \in \Sigma^*$ define $b_{\Sigma}(x)$ to be its binary encoding.

For $x, y \in \Sigma^*$, to encode the pair of $x$ and $y$ can add as $x, y$ over $\Sigma' = \Sigma \cup \{,\}$.

Or sometimes better: $(x, y) := 0|b_{\Sigma}(x)|1 b_{\Sigma}(x) b_{\Sigma}(y)$
TM Encoding

Can encode a TM as a bit string:
- n (states), m (tape symbols), (first) k (are input symbols),
- s (start state), t (accept state), r (reject state), u (blank symbol),
- transition1, transition2, …

( (p, i), (q, j, L) ), ( (p, i), (q, j, R) ) , …

Similarly, we can encode DFAs and NFAs as bit strings
Other ways to encode a TM exist:

\[ 0^n10^m10^k10^s10^t10^r10^u1 \ldots \]

- \( n \) states
- \( m \) tape symbols (first \( k \) are input symbols)
- \( s \) start state
- \( t \) accept state
- \( r \) reject state
- \( u \) blank symbol

\[
( (p, i), (q, j, L) ) = 0^p10^i10^q10^j10
\]

\[
( (p, i), (q, j, R) ) = 0^p10^i10^q10^j100
\]
Binary languages about computations

Define the following languages over \{0,1\}:

\( A_{\text{DFA}} = \{ (B, w) \mid B \text{ encodes a DFA over some } \Sigma, \)
\( \text{ and } B \text{ accepts } w \in \Sigma^* \} \)

\( A_{\text{NFA}} = \{ (B, w) \mid B \text{ encodes an NFA, } B \text{ accepts } w \} \)

\( A_{\text{TM}} = \{ (M, w) \mid M \text{ encodes a TM, } M \text{ accepts } w \} \)
\[ A_{TM} = \{ (M, w) \mid M \text{ encodes a TM over some } \Sigma, \]
\[ w \text{ encodes a string over } \Sigma \text{ and } M \text{ accepts } w \} \]

Technical Note:
We’ll use an decoding of pairs, TMs, and strings so that every binary string decodes to some pair \((M, w)\)

If \(x \in \{0,1\}^*\) doesn’t decode to \((M,w)\) in the usual way, then we define that \(x\) decodes to the pair \((D, \varepsilon)\)

where \(D\) is a “dummy” TM that accepts nothing.

Then, we can define the complement of \(A_{TM}\) very simply:

\[ \neg A_{TM} = \{ (M, w) \mid M \text{ does not accept } w \} \]
Universal Turing Machines

Theorem: There is a Turing machine $U$ which takes as input:
- (1) the code of an arbitrary TM $M$
- (2) an input string $w$
such that $U$ accepts $(M, w) \iff M$ accepts $w$.

This is a fundamental property of TMs: There is a Turing Machine that can run arbitrary Turing Machine code!

Note that DFAs/NFAs do not have this property: $A_{DFA}$ and $A_{NFA}$ are not regular.
$A_{DFA} = \{(D, w) \mid D \text{ is a DFA that accepts string } w\}$

**Theorem:** $A_{DFA}$ is decidable

**Proof:** A DFA is a special case of a TM. Run the universal $U$ on $(D, w)$ and output its answer.

$A_{NFA} = \{(N, w) \mid N \text{ is an NFA that accepts string } w\}$

**Theorem:** $A_{NFA}$ is decidable. (Why?)

$A_{TM} = \{(M, w) \mid M \text{ is a TM that accepts string } w\}$

**Theorem:** $A_{TM}$ is recognizable but not decidable!
Parting thoughts:
Everything is zeros and ones, even Turing Machines. Questions about computations are natural and important. Universal TM – separating hardware from software.