A TM accepts a string if, given the string as input, the TM reaches $q_{accept}$.

A language is recognizable (recursively enumerable) if there is a Turing machine that accepts exactly the strings in the language.

A language is decidable (recursive) if it is accepted by a TM that always halts (i.e., if the TM always ends up in $q_{accept}$ or $q_{reject}$.)
Rather than accepting a language, we can use a TM to compute a function:

- The machine starts with some string $x$ on its tape initially.
- The machine halts with some string $y$ on its tape finally.
- The corresponding function $f$ would have

$$f(x) = y$$

- If the TM doesn’t halt, it computes a partial function.
Several variant definitions. Each specify a language L.

1. A TM that prints out all the members of L, one at a time (but not necessarily in any particular order)

2. A TM that prints out all the members of L, one at a time (but...) with arbitrarily many repeats.

3. A TM that, given an integer n, returns the n\textsuperscript{th} element of a sequence like (1) above.

4. A TM that, given an integer n, returns the n\textsuperscript{th} element of a sequence like (2) above.

For a fixed language, all these are interconvertable.

A language is recognizable iff it can be enumerated.
Encoding TMs as Strings

We know that a TM can be described fully by a finite list of transition rules, of the form:

- $q_0, \_ \rightarrow q_i, \_, R$
- $q_i, \_ \rightarrow q_j, \_, L$
- $q_i, x \rightarrow q_k, x, L$
- $q_m, a \rightarrow q_n, x, R$ etc.

We can devise a way to encode an arbitrary set of such rules into a fixed alphabet.

Although the number of states and tape symbols can be arbitrary we can encode these by using strings of symbols, say \{0, 1\}

concatenate the symbols to describe rules

concatenate rules to describe machines.
Encoding TMs as Strings

- Not every string of symbols in the encoding alphabet must correspond to a well-formed TM description.
- But we can determine algorithmically which are and which aren’t.
  - In fact, a TM can do so!
  - For those that aren’t, assume they describe a default TM that immediately halts and rejects.
- We thus can enumerate all TMs:
  \[ M_0, M_1, M_2, \ldots \]
- We can do the same for encodings of initial tapes:
  \[ x_0, x_1, x_2, \ldots \]
- We can even enumerate all pairs: \( \langle M_i, x_j \rangle \)
An encoding of a Turing machine can be viewed as a program.

A Turing machine that interprets such a program to carry out the actions specified is called a Universal Turing Machine (UTM).
Universal Turing

UTMs can be shown to exist by constructing them.

Think about what would be required.

- The tape has to hold the tape of the machine being simulated.
- The tape has to hold the program of the machine being simulated.
- The program must be laid out in such a way that the necessary markers can be inserted to keep track of the current state, etc.

All this is possible, if somewhat laborious to construct.

Whether a machine is universal will depend on the particular encoding used.
Specific UTMs

- The first was constructed by Turing himself.
- Shannon showed any UTM could be converted either to a 2-symbol machine or to a 2-state machine by augmenting states or symbols, respectively.
- Minsky (1960) gave a 7-state 6-symbol machine.
- Watanabe (1961) gave an 8-state 5-symbol machine.
- Minsky (1962) gave a 7-state 4-symbol machine.
- Rogozhin (1996) gave a 4-state 6-symbol machine.
- Wolfram and Reed (2002) gave a 2-state 5-symbol machine.
- Smith and Wolfram (2007) gave a 2-state 3-symbol machine.
- No 2-state 2-symbol UTM exists.
A Specific Universal Turing Machine

2-state, 5 symbol UTM published by Wolfram in 2002

Symbols and states

Computability and Uncomputability
Question

How is my laptop more like a Finite State Machine than like a Turing Machine?

How is my laptop more like a Turing Machine than like a Finite State Machine?
Church-Turing Thesis

✓ If it can be done at all, then it can be done by
  ✓ A Turing Machine
  ✓ Lambda Calculus
  ✓ An Unrestricted Grammar
  ✓ A 2-register machine
  ✓ C
  ✓ ...

✓ Note: assumes suitably coded inputs and outputs
Is There More?

- Regular: $a^*b^*$
- Context Free: $a^n b^n$
- Decidable: $a^p b^p c^p$ (p perfect)
Some Languages Aren’t Decidable

Given a finite $\Sigma$, how many languages are there?

How many TMs are there?

QED
But wait...

- How many languages over $\Sigma$ could I describe (say, in a LaTeX document)?
- How many TMs are there?
- QED?
Languages of Acceptance

Which are recognizable (by a TM)? Decidable?

- $A_{DFA} = \{ \langle D, w \rangle \mid D \text{ a DFA, } D \text{ accepts } w \}$
- $A_{NFA} = \{ \langle N, w \rangle \mid N \text{ an NFA, } N \text{ accepts } w \}$
- $A_{RE} = \{ \langle R, w \rangle \mid R \text{ a regexp, } R \text{ matches } w \}$
- $A_{CFG} = \{ \langle G, w \rangle \mid G \text{ a CFG, } G \text{ produces } w \}$
- $A_{TM} = \{ \langle M, w \rangle \mid M \text{ a TM, } M \text{ accepts } w \}$
Digression: Bootstrapping a Compiler

✓ Lots of compilers are written in the same language they compile!
   ✓ Gnu C Compiler (used in CS 105) is written in C
   ✓ Glasgow Haskell Compiler (used in CS 131) is in Haskell
   ✓ etc.

✓ Consequence:
   ✓ We sometimes run programs (compilers) on their own source code!
$A_{TM}$ is not decidable

<table>
<thead>
<tr>
<th></th>
<th>$\langle M_0 \rangle$</th>
<th>$\langle M_1 \rangle$</th>
<th>$\langle M_2 \rangle$</th>
<th>$\langle M_3 \rangle$</th>
<th>$\langle M_4 \rangle$</th>
<th>$\vdots$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_0$</td>
<td>Acc.</td>
<td></td>
<td>Acc.</td>
<td></td>
<td>Acc.</td>
<td></td>
</tr>
<tr>
<td>$M_1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_2$</td>
<td>Acc.</td>
<td>Acc.</td>
<td></td>
<td>Acc.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\vdots$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Is There More?

Regular: $a^*b^*$
Context Free: $a^n b^n$
Decidable: $a_p b^p c^p$ (p perfect)
Recognizable: $A_{TM}$
Recall:

- If $L$ and $L^c$ are both recognizable, then they are both decidable.

- What is the complement of $A_{TM}$?
Recognizable
Decidable
Context
Free
Regular
\(a^*b^*\)
\(a^n b^n\)
Decidable
\(a^p b^p c^p\)
(p perfect)
Recognizable
\(A_{TM}\)
\((A_{TM})^c\)
Obligatory Corollary

The language

\[ H = \{ \langle M, w \rangle \mid M \text{ a TM that halts given } w \} \]

is not decidable.

Proof: Suppose there were a halt-checking TM....