Recall:

- Turing machine

**Def. Algorithm = Halting Turing machine**

**Def. A language is**
- Turing recognizable/recursive if it is the language of a Turing machine
- Decidable/recursive if it is the language of a halting Turing machine (algorithm)

The Turing machine model is
- powerful (can simulate PDAs, general-purpose computers), and
- robust (modifying the model, e.g., allowing multiple tracks or multiple tapes, does not change its power).

**Exercise:** Explain how a standard Turing machine can simulate a 2-dimensional "tape" on which read/write head can move up/down/left/right.

**Exercise:** Show that a single-tape T.M. that cannot overwrite the input string can only recognize regular languages.

**Today:** Universal Turing machine - can simulate all other TMs

Undecidable problems
Why "recursively enumerable"?

**Theorem:** A language is recursively enumerable if and only if some enumerator prints a list of all the strings in the language.

**Proof:**

**Easy direction \( \Leftarrow \):** Assume we have an enumerator that enumerates the strings of \( L \). Let \( M \) be the following Turing machine:

\( M: \) On input \( x \):

1. Run the enumerator. Every time it prints a string, compare it to \( x \).
2. If \( x \) ever appears, accept.

**Harder direction \( \Rightarrow \):** Let \( M \) be a Turing machine for language \( L \). Enumerator \( E \):

For \( k = 0, 1, 2, \ldots \):

- Simulate \( M \) for \( k \) steps on each string in \( \Sigma^* \).
- If any computations accept, print the string.
- Any string that is eventually accepted will eventually be printed.
- Effectively, parallel simulation of \( M \) on all possible inputs.

**Exercise:** A language is decidable if and only if some enumerator prints it out in lexicographic order.
Example: The set of decidable languages is closed under:
- union
- concatenation
- star
- intersection
- complementation.

Example: The set of recursively enumerable languages is closed under:
- union
- concatenation
- star
- intersection.

Lemma: $L_1, L_2$ decidable
$\Rightarrow L_1 \cup L_2$ decidable.

Proof:
Simulate the two algorithms in series. When $A_i$ halts in a non-final state, clean up the tape and transition to $A_0$'s start state.$\square$

Lemma: $L_1, L_2$ recursively enum.
$\Rightarrow L_1 \cup L_2$ is r.e.

Proof:
Simulate the two Turing machines in parallel. Accept if either accepts.$\square$

Lemma: $L$ decidable $\Rightarrow \overline{L}$ decidable.

Proof:

Lemma: $L, \overline{L}$ rec. enum.
$\Rightarrow L$ decidable.

Proof:

Corollary: For a language $L$, either
- both $L, \overline{L}$ are decidable/recursive, or
- neither $L$ nor $\overline{L}$ is rec. enum., or
- one is rec. enum., but not decidable, other is not rec. enum.

<table>
<thead>
<tr>
<th>Decidable</th>
<th>rec. enum.</th>
<th>not decidable</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>
Universal Turing machine

Definition: Universal language
\[ L_U = \{<M, \omega> | M \text{ is the encoding of a Turing machine, } \omega \text{ is an input, and } \omega \in L(M) \} \]

Exercise: Specify some Turing machine encoding/programming language.
(The details are not so important)

Example answer: Let \( M \) be a Turing machine, with states \( Q = \{q_1, q_2, \ldots, q_n\} \). Assume (without loss of generality) \( q_1 \) is initial state, \( F = \{q_3\} \).
Let \( \omega_1 = \varepsilon \) and \( \omega_2, \ldots, \omega_k \) index tape alphabet \( \Gamma \).

\begin{align*}
\text{Encoding} & \quad 0^410^210^110 \\
\text{Transition} & \quad \sigma(q_i, \omega_1) = (q_k, \omega_2, \lambda) \\
\text{Encoding} & \quad 0^410^210^110^2100 \\
\text{Transition} & \quad \sigma(q_i, \omega_2) = (q_k, \omega_2, \text{right})
\end{align*}

\[ <M> = 11 \text{<code>} 11 \text{<code>} 11 \ldots \]

Encodings of transitions

Note: Turing machines have multiple encodings.
Not every string is a valid encoding.

Theorem: The universal language \( L_U \) is recursively enumerable.
That is, there is a Turing machine \( V \) such that \( L_U = L(V) \).

(Morally, "Microsoft Theorem": General-purpose computers exist, don't need specific hardware devices for every problem, use software instead)

Proof sketch:

High-level description: On input \( <M, \omega> \) for a T.M. \( M \) and string \( \omega \), simulate \( M \) on input \( \omega \), and accept if \( M \) ever enters an accepting state.

Implementation: Use four tapes:
1. contains \( <M> \) (read only)
2. current state \( q_i \) of \( M \)
3. for simulating \( M \)'s tape
4. scratch space

Transitions are straightforward... [HMU 9.2.8]

Note: This simulation is efficient (linear time). If it took exponential time, the software industry would not exist.
Theorem: The universal language $L_u$ is undecidable.

Proof: Assume, for contradiction, that $L_u$ is decidable.
Let $V$ be a halting Turing machine for $L_u$.
Let $W$ be as follows:
On input $\langle M \rangle$ the encoding of a Turing machine $M$:
1. Run $V$ on input $\langle M, \langle M \rangle \rangle$
2. Accept if and only if $V$ rejects.

Consider giving $W$ its own encoding.
$\langle W \rangle \in L(W) \iff \langle W, \langle W \rangle \rangle \notin L(V)$ [def of $W$]

$\iff \langle W \rangle \notin L(W)$ [def. of $V$]
a contradiction!

The main idea of this proof is diagonalization [EHMU 9.17]