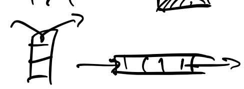
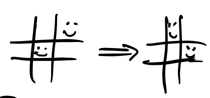
COMS W3261 — Lecture 9, part 3:

From Turing Machines to Algorithms. Church-Turing Thesis

- We saw: several models of computation are equivalent in power to TMs:
  - Multitape TMs.
  - Nondeterministic TMs.
- What other high-level automata exist?
  - Post-Turing Machine (Post, 1936). = TM
  - Lambda Calculus (Church, 1930s). = TM
  - Wang Tile (1961). = TM 
  - Automaton with a Queue.  = TM

(Lots of things are equivalent!)

- Anything that can simulate a TM can be used to recognize and decide languages. (This property is called Turing-completeness.)
  - Most programming languages (C, C++, Java, Python, JavaScript, ...)
  - LATEX
  - Conway's Game of Life. 
  - Microsoft Excel and PowerPoint.
  - Minecraft, Portal, Magic the Gathering...

☆

$$\boxed{\text{The Church-Turing thesis: "Our intuitive notion of an algorithm (completely specified process for performing a task) corresponds precisely to the set of tasks that can be performed on a Turing machine."}}$$

☆

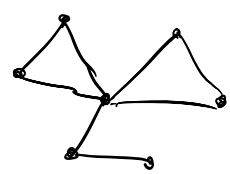
Encoding Problems: How do we speak generally, but precisely, about powerful automata?

<u>Ideal</u>: Every high-level description should be reducible to a full formal description (in principle.)

<u>So far</u>:
- Formal descriptions (7-tuples)
- Implementation descriptions (Describe head movement and tape management)
- High-level description: precise English prose that describes an <u>algorithm</u> while ignoring implementation details.

<u>Observation</u>: All finite mathematical objects can be encoded as strings.

<u>Example</u>. $G = (V, E)$



We can write G as a string over any (non-empty) alphabet using an unspecified encoding as $\underbrace{\langle G \rangle}_{G\text{, encoded as a string.}}$.

Example: could list all vertices and edges.

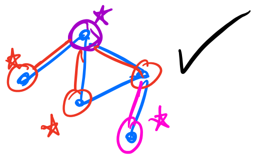
- We will assume that if some TM takes an encoded object $\langle O \rangle$ as input, it begins by scanning $\langle O \rangle$ and rejecting if $\langle O \rangle$ is not a valid encoding of the right kind of object.

Example. Recognize $A = \{ \langle G \rangle \mid G \text{ is a connected, undirected graph.} \}$

High-level description of a TM for A:

$M =$ "On input $\langle G \rangle$, an encoding of $G = (V, E)$:

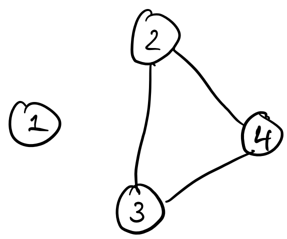
(0. Reject if $\langle G \rangle$ is not an encoded graph)
1. Select a node of G and mark it.
2. For each node, mark it if it is adjacent to some marked node.
3. Repeat (2) until no new nodes are marked.
4. If all nodes are marked, accept; otherwise, reject."



How does this translate to implementation details?

Well – suppose

G:



Could encode G as (vertices)(edges):

"(1, 2, 3, 4)((2,3)(3,4)(2,4))"

Step 2:

For each marked node u:

For each unmarked node v:

Check if $(u,v) \in E$ and mark v if so.

Next time: More high-level descriptions of TMs, show many cool languages are/are not decidable.

Reading: Sipser 3.1 (TMs)

Sipser 3.2 (Variant TMs)

Sipser 3.3 (High-level representations, CT thesis).