#1 Encodings and the UTM—Motivation

- 1. Real problems are usually about things like numbers, graphs, trees, and so forth. We will use TMs to describe algorithms that solve real problems, so we need a way to represent these objects.
- 2. A Turing machine has components that are each representable. Therefore, we can consider a Turing machine to be such an object.
- 3. What questions can be asked about Turing machines by Turing machines? This line of reasoning starts our trek into undecidability.

#2 Some encodings

We will generally stick to $\Sigma = \{0, 1\}$. This is sufficient to represent basically everything.

Example 1 (Binary Numbers). The numbers you are familiar with exist in a representation called a decimal expansion (deci meaning base ten). So, for example,

$$1234 = 1 * 10^3 + 2 * 10^2 + 3 * 10^1 + 4 * 10^0$$

A binary expansion changes the base from 10 to 2. For example, 6 in binary is 1110

$$1110 = 1 * 2^3 + 1 * 2^2 + 1 * 2^1 + 0 * 2^0$$

Example 2 (Integers). Integers are similar, except that we use the leading digit to represent negativity. So 1110 is now

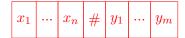
$$1*(-1) + 1*2^1 + 1*2^1 + 0*2^0$$

Let us move onto higher mathematical constructs.

Example 3 (Pairs and tuples). As we have seen in class, if I have $x, y \in \Sigma^*$, then I can represent the pair $(x, y) \in \Sigma^* \times \Sigma^*$ with a delimiter. Let's use #.

$$x \# y$$

On the input tape, suppose $x = x_1...x_n$ and $y = y_1...y_m$:



Example 4 (Graphs). Consider this definition.

Definition 1 (Graph). A graph G = (V, E) is a 2-tuple where V is a set of vertices and $E \subseteq V \times V$ is a set of edges on those vertices.

Let us think in class how we might encode this when $V \subseteq \mathbb{N}$. I can think of two ways: I suspect that class will think of encoding V as a delimited string of naturals and then E as a delimited string of pairs. Another way to do it is by encoding the transition matrix.

In general, given some mathematical object x (such as a graph, pair, number, etc) we denote its encoding as a string as $\langle x \rangle$. For example, we might write $\langle 2 \rangle = 10$ in binary and $\langle (x,y) \rangle = x \# y$.

#3 TMs with encoded input

The power of encodings permits us to start writing Turing machines that answer questions about more interesting objects. An example.

Here is a classic graph problem called the independent set problem, or INDSET. In plain english, INDSET asks: given a graph G and a number k, is there a k-size independent subset of G's vertices? We would write formally:

Example 5.

INDSET =
$$\{\langle G, k \rangle : \exists S \subseteq V(G) \text{ s.t. } |S| \geq k \text{ and } \forall u, v \in S, (u, v) \notin E\}$$

What does a Turing machine that recognizes this language look like? What is the format of its input? Give me the broad idea here.

#4 Encoding automata

Recall the definition of a DFA D.

$$D = (Q, \Sigma, \delta, q_0, F)$$

Can we encode this? Yes. So we can write Turing machines that answer question about DFAs.

$$A_{DFA} = \{\langle D, w \rangle \mid D \text{ accepts input string w}\}$$

Further, this language is decidable. It is a good exercise to think about how. See Sipser Theorem 4.1 for a full writeup.