#1 Motivation

1. Single tape TMs are simpler to describe but often a pain in the ass. Much easier to work with many tapes.
2. Another style of Turing Machine is to think less about deciding languages and more about TMs that *compute functions*, which is easier to express with two tapes.

#2 Multitape TMs

#2.1 Definition

Definition 1. *A Multitape TM is a Turing machine with all components equal except that δ has the type signature*

$$\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, S\}^k$$

where k is the number of tapes.

The expression

$$\delta(q_i, a_1, ..., a_k) = (q_j, b_1, ..., b_k, D_1, D_2, ..., D_k)$$

means that if the machine is in state q_i and heads 1 through k are reading symbols a_1 through a_k, then the machine transitions to state q_j, writes symbols b_1 through b_k on takes 1...k and directs heads 1..k to go in direction D_1...D_k.
(This is the same idea as for one tape but we now specify what *many* tapes do on each input.)

#2.2 Usage

Let $\Sigma = \{0, 1\}$ and consider the language A defined by

$$L = \{w \mid w \text{ contains an equal number of 0s and 1s}\}$$

How might we solve this using multiple tapes? I propose something like this:

1. Write all 0's onto work tape 0.
2. Write all the 1's onto work tape 1.
3. Check if work tapes 1 and 2 are of equal length.

(Let's try an example in class, like 011010).

#2.3 Equivalence (Proof Sketch)

Claim 1. *Every multitape TM M has an equivalent TM M'.*

Proof Sketch. To simulate many tapes, store each work tape on the input tape delimited by the # symbol. To simulate many heads, build a new $\Gamma' = \Gamma \cup \{a' \mid a \in \Gamma\}$ where the symbol a' is "like" the symbol a except that we are pretending one of the work-tape heads is at that symbol (call this a *virtual head*). For example, consider input $w = w_1...w_k$. Then we have starting configuration:

$$\#w_1'w_2...w_k\#\square'\#\square'\#...\#$$

Next, simulate one step of the multitape machine as so.

1. Scan the tape from the first # to the last, making note of which symbols are under each virtual head.
2. Scan back, updating each work-tape segment as per how the transition function of M dictates.
3. Repeat. If at some point a virtual head transitions to the right of an #, shift the remaining rightmost cells over by 1 to make room.

□

#3 Example Multitape TM: Computing Functions, *or* Membership versus computation

The following definition is a convenient formulation that I couldn't find formally anywhere, but in my opinion is maybe the easiest TM to work with.

Definition 2 (Computing TMs)**.** *A* Computing Turing machine M *is a multitape TM with*

1. *at least two distinguished tapes: an* input tape *and an* output tape*, and*
2. *only one halting state q_H.*

We say the M computes *y on input x, written $M(x) = y$, if the input x transitions M to the halt state with y on its output tape.*

Definition 3 (Computable Functions)**.** *A function $f : \Sigma^* \to \Sigma^*$ is a* computable function *if there exists Turing machine M that, on every input w, halts with just $f(w)$ on its output tape.*

#3.1 Equivalence (Proof Sketch)

The notions of "computing functions" and of "deciding languages" are actually equivalent.

Case: Computing functions → deciding languages. Suppose f is a computable function computed by M_f. For each $x \in \Sigma^*$, we can equivalently consider the language

$$F = \{(x, y) | f(x) = y\}$$

and then use M_f to build some TM M that decides F. Note that we can encode pairs easily on a TM by using a delimiter, e.g., the string $x\#y$. (We will talk more on encodings later.)

Case: Deciding Languages → computing functions. Suppose M is a TM that decides the language A. Membership in A can be thought of as a *boolean relation* $f : \Sigma^* \to \{0, 1\}$, where $f(x) = 1$ if M accepts x and $f(x) = 0$ if M rejects x. So, use M to build some M_f that computes such a function.