CS 360
Fall 2021
Doing More Operations with Turing Machines

Jeffrey Shallit
School of Computer Science
University of Waterloo
shallit@uwaterloo.ca
https://cs.uwaterloo.ca/~shallit
Just like in programming, we can use “subroutines” or “helper methods” on Turing machines.

These are Turing machines that assume the tape configuration is in some certain format, and carry out a transformation on the symbols of the tape.

It is important, when designing such subroutines, to explicitly state what the initial and final configurations that you expect are (including position of the tape head).

So study the following examples until you can easily do them yourself, and they become obvious and simple for you.
Let's see in detail how to design a subroutine for a basic operation: copying the contents of tape 2 to the “end” of tape 1.

We used this in the previous lecture.

Here we will assume that the “end” of tape 1 is marked by the last non-blank character.

We’ll also assume that for both tapes, we start scanning the blank before the input.

And that at the end of the computation, we return to that blank.
Copying on a 2-tape TM

Transitions labeled \((a_1, a_2), (b_1, b_2)(d_1, d_2)\) means replace \(a_1\) with \(b_1\) on tape 1 and move in direction \(d_1\), and replace \(a_2\) with \(b_2\) on tape 2 and move in direction \(d_2\).

In state \(q_1\) we walk right on tape 1 to get to the “end”.

In state \(q_2\) we walk right on both tapes, copying the symbol from tape 2 to tape 1.

In state \(q_3\) we “rewind” tape 1 and in state \(q_4\) we “rewind” tape 2.
Now let's look at a related subroutine: copying tape 1 reversed to tape 2.

We assume tape 1 initially holds a string and tape 2 is initially blank.

In state $q_1$ we walk to the “end” of tape 1. In state $q_2$ we walk left on tape 1, and right on tape 2, copying each symbol from tape 1 to tape 2. In state $q_3$ we walk right on tape 2 back to the beginning.
Comparing two tapes

Imagine that tape 1 holds some string and tape 2 holds some string. We want to decide if the contents of the tapes are the same (up to the second blank). We can do this as follows:

In state $q_1$ we move right as long as we see two identical symbols. If we hit two blanks we know the inputs are the same. If we hit two different symbols we know the inputs agreed up to now, but are now different. In both cases we do not change the contents of the tape.
So far we have seen how to use a TM to recognize languages like 
\( \{ a^n b^n : n \geq 0 \} \), and \( \{ a^{2^n} : n \geq 0 \} \).

Now let’s see how to do a common operation in a programming language, like integer multiplication.

This could be used in a subroutine to do a number theory calculation, for example.

We will use the unary encoding of integers, representing the natural number \( n \) by the string \( a^n \).

So, given inputs of \( a^m \) and \( a^n \) we want to calculate \( a^{mn} \).
Let us assume that we have a 3-tape machine.

The first tape initially holds $a^m$.

The second tape initially holds $a^n$.

When the calculation is complete, we want tape 3 to hold $a^{mn}$.

The easiest way is to copy the contents of tape 1 over and over to tape 3, $n$ times.

Each time we copy the contents of tape 1 to the end of tape 3, we change one of the $a$'s on tape 2 to an $X$.

When we reach the end of the $a$'s on tape 2, we then “rewind” all tapes back to the blank.