Variations of Turing Machines

*We show that changing the model,
making it less or more restrictive,
does not change the power of a TM.*

TMs as Transducers

A TM that perform calculations is a ***transducer***. It leaves the answer on the tape.

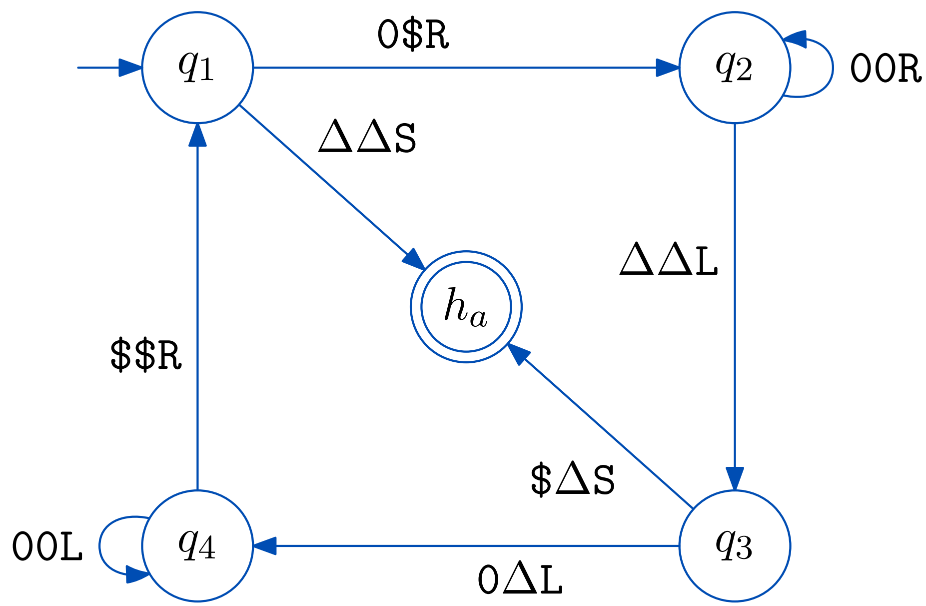
For example, a TM that starts with $\$^i\#\$^j$ on tape and ends with $\$^{ij}$ does multiplication.

Example: Unary Halving

A TM that treats input as unary number and divides it by 2.

Example: Unary Halving

It changes first symbol to another symbol, and then deletes the last symbol. And repeats. (At end, we should revert new symbol to old.)



A T-computable Function

A function f that converts strings into strings is ***T-computable*** if some TM M computes it.

That is, M always halts, and on input w, M halts with $f(w)$ on its tape.

Variations on the Model

The definition of a Turing Machine is robust: Many variations do not alter its power.

The general idea is:

- If capability is added, then show that standard TM can simulate it.

- If capability is removed, then show that crippled TM can simulate standard one.

Example: Omitting Stay-In-Place Option

For example, suppose we force TM to move its head each time.

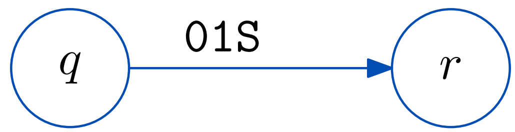
Well, one can achieve the net effect of stay-in-place by moving the head off the cell and immediately moving it back!

How does one ensure the head moves back?

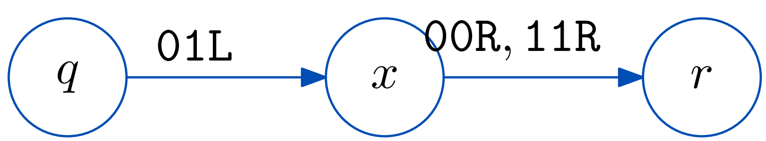
How Does One Ensure Head Moves Back?

Move to new intermediate state!

For example, transition $\delta(q, 0) = (r, 1, S)$ becomes $\delta(q, 0) = (x, 1, L)$, and $\delta(x, ANY) = (r, R, ANY)$, where x is new state:



becomes



Example: Medusa

Call a multi-headed TM the ***Medusa***.

A standard TM can simulate the Medusa by storing the location of the Medusa's heads. For example, the standard TM could represent each Medusan head by a new symbol $\#_1$, $\#_2$, etc.:

Example: Medusa Simulation

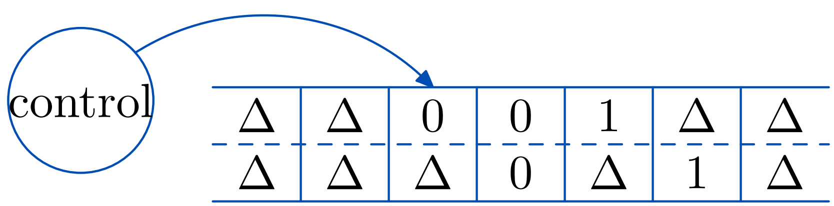
To simulate a step of the Medusa, the standard TM sweeps along its tape, finds each Medusan head, and updates it.

Note that the important thing is simulation, not the number of steps.

Multiple Tracks

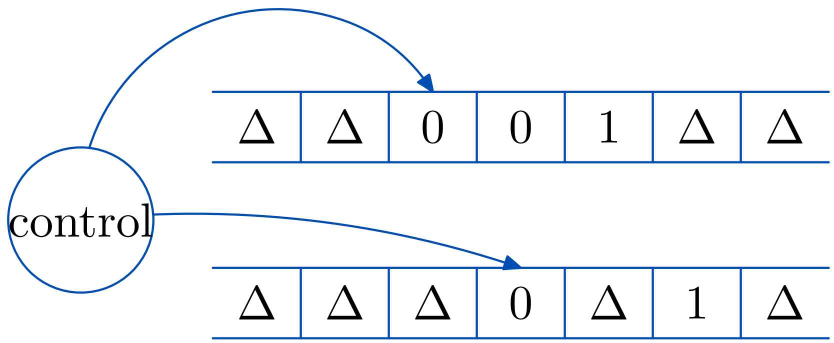
A ***2-track TM*** is one where there are two symbols in each cell, an upper one and a lower one.

One way to simulate this, is to create a new alphabet: each letter of the alphabet represents a pair of symbols.



Multiple Tapes

A TM with multiple tapes has the same power as a standard TM.



One approach is to convert a multitape TM to a multitrack TM, storing the positions of the heads as in the Medusa.

Nondeterminism

Nondeterminism means that the TM may have more than one choice of action. As usual, a nondeterministic TM (or ***NTM***) accepts a string if some choice of actions lead to the accept state.

Theorem. *A nondeterministic TM has the same power as a standard TM.*

Proof Idea

We show that the NTM can be simulated by a deterministic one. Well, we try all possible choices!

We need the concept of ***configuration***. This is a record of the complete status of a TM: its state, tape contents, and head position. (Note only finite portion of tape is used at any stage.)

Proof of Theorem

We view the calculations of NTM as a *tree*. The nodes are the configurations of the NTM, and the children of a node are the possible next steps. The NTM accepts the input if there is a branch that leads to an accepting configuration.

The simulator does *breadth-first-search* of tree.

A TM Can Simulate a Computer

At first, a TM appears primitive. But one can show that one can use the first tape as ***random access memory***, as in a normal computer, if second tape has address.

Further, one can show that one can translate any program for a normal computer into a program for a TM:

Fact. *A Turing Machine can simulate a real computer.*

Church's Thesis

Several models of computation have been proposed over the years, but they have exactly the same power as a TM as recognizers:

Church's "thesis" is the belief/claim that the model is appropriate and has all the power of *any* computer we might build.

> **Church's thesis.** *There is an "effective procedure" for a problem if and only if there is a TM for the problem.*

Universal TMs

A ***universal TM*** is a TM that takes another TM as an input. For this, one needs to specify an encoding of a TM. Universal TMs have been devised with surprisingly few states.

Related Models

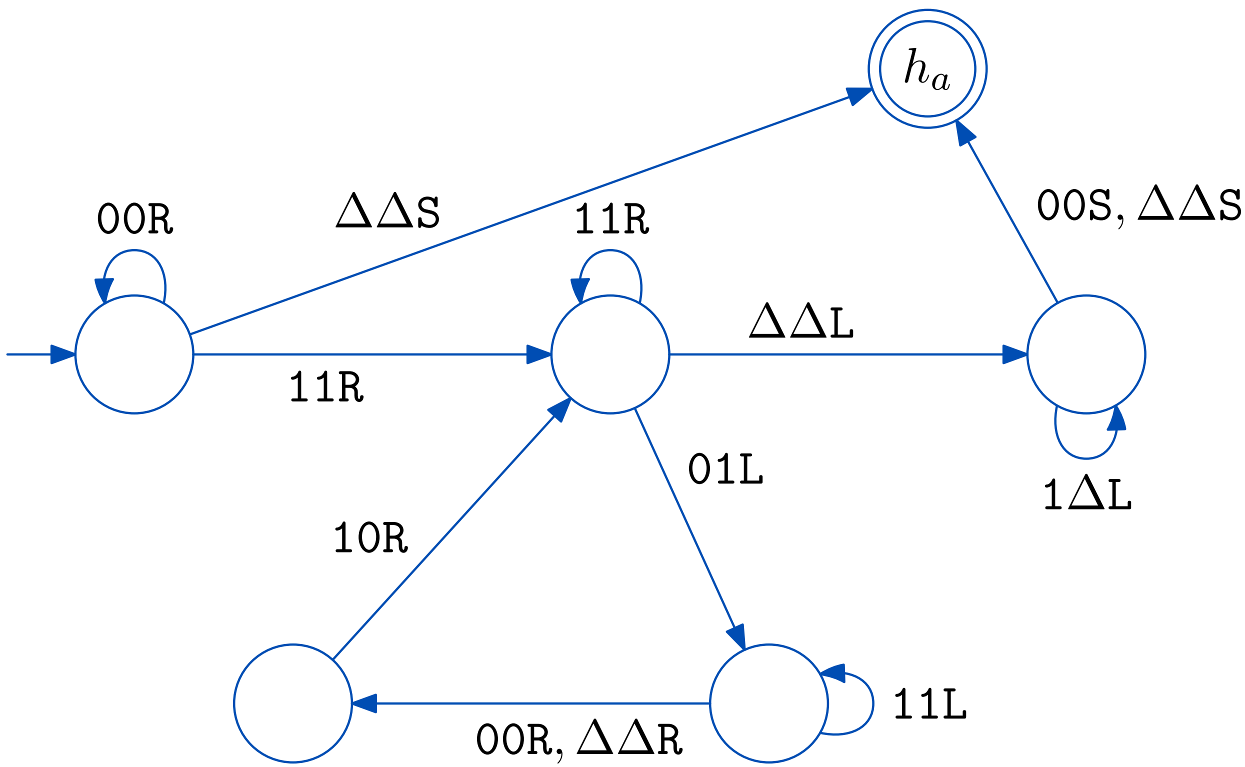
The exercises consider connections between TMs and other machines, including ones with multiple stacks or ones with a queue.

Practice 1

Draw a TM that erases all instances of a certain symbol from the input. Say the alphabet is $\{0, 1\}$ and the TM erases all 1's. For example, if input is 10101100, output is 0000.

Solutions to Practice 1

The idea is to move each 0 to the left; then erase the 1's.



Practice 2

A Jittery TM is one that always writes a different symbol to the one it has just read. Show that a Jittery TM can simulate a standard TM.

Solutions to Practice 2

For each symbol in Γ, add a copy. Then for each move of the standard TM, the Jittery TM makes two moves: it first writes the duplicate symbol, staying put but going to a temporary state; then it writes the real symbol and moves to the correct state.

For example, the transition $\delta(q, \mathtt{0}) = (r, \mathtt{0}, \mathtt{L})$ becomes $\delta(q, \mathtt{0}) = (q', \mathtt{0}', \mathtt{S})$ and $\delta(q', \mathtt{0}') = (r, \mathtt{0}, \mathtt{L})$.

Summary

A normal TM can simulate a TM with a one-way infinite tape, with multiple tapes, and so forth. A nondeterministic TM is no more powerful than a normal one. Church's thesis says that there is an algorithm for a problem if and only if there is a TM for it. A TM can simulate a normal computer. A universal TM is one that can execute any other TM as an input.