COSC 341
Theory of Computing
Lecture 13
Powering up Turing Machines

Stephen Cranefield
stephen.cranefield@otago.ac.nz

Lecture slides (mostly) by Michael Albert

Keywords: Multi-track, multi-tape,
non-determinism, oracles

Why do we want more powerful machines?

The Church-Turing thesis says that, once we reach the power of Turing machines, the model doesn't matter any more.

So what's the point of extending the model of mechanical computing beyond Turing machines?

- To instil confidence that the Church-Turing thesis is correct.
- When showing that a problem is mechanically solvable we can use a more powerful and convenient extended machine.
- When showing that a problem is not mechanically solvable we can restrict ourselves to ordinary TMs.

Multiple tracks

In a multi-track Turing machine:

- Instead of one tape we have several.
- There is still a single read-write head which simultaneously reads aligned symbols on the various tapes.
- It can write on them all before moving in the same direction everywhere.

This is really just an ordinary TM with an extended alphabet (if there are k tracks the new alphabet has as its letters the k-tuples of letters from the original alphabet).

Two-way tape

What about the tape that's infinite in both directions? In some defintions of a TM, that's the standard model; indeed in Anthony Morphett's TM simulator you have to set an option to make it one-way.

- Simulate such a machine with a two-track machine.
- The bottom track represents the "positive" part of the tape and the top track the "negative" part.
- States of the old machine need to be copied twice - once to indicate "pay attention to the symbol in the 'positive' track" and once for the 'negative' track (directions adjusted appropriately).
- Passing through 0 also needs a little bookkeeping work.

Multiple tapes

In a multi-tape Turing machine:

- There are a number of separate tapes.
- Each tape has its own read-write head and these can move independently of one another.
- Transitions are based on the complete sequence of symbols read at any one time, as well as the current state.
- That is, there is one central control unit which takes charge of all the different heads simultaneously.

Simulating one of these using one of the previous examples is a little tricky!

Simulating a k-tape machine

Use a multi-track machine with $2k + 1$ tracks.

- One has a $\#$ in the leftmost cell and nothing else. Its only purpose is to allow us to reliably wind the head back to the leftmost cell.
- For each of the k tapes of the multi tape machine we dedicate two tracks.
  - One contains the actual contents of the tape.
  - The other contains a single symbol M which marks where the 'virtual' version of the read-write head for that tape is currently located.

Simulating a single step of the multi-tape machine requires a long sequence of steps . . .

Simulating a k-tape machine (continued)

- Always assume that the read write head begins at the leftmost cell.
- One by one we advance to the various marked cells and determine the symbols on each tape, storing these by means of state.
- Then rewind back to the left again.
- Now we know from the original transition table of the multi-tape machine what should be written on each tape and which way its head should be moved.
- Again, advance to each of the marks, change the symbol on the appropriate track, and move the mark as necessary.
- When all this is done, go back to the left (remembering the new state), and start again.

Non-determinism

Non-determinism for Turing machines comes in two flavours:

- Non-determinism by transition (NDT)

  Multiple possible transitions are allowed from a given state/symbol pair. A word w is accepted if there is some sequence of choices of allowed transitions that halts and accepts.
- Non-determinism by oracle (NDO)

  A deterministic two-tape machine. On one tape we write the actual input w. On the other, an oracle tape, a genie writes something that will be helpful for verifying whether w is in the language. We then run the machine and accept w if it halts and accepts. The accepted language is exactly the set of strings which we accept for *some* choice that the genie can make.

And they're the same

Theorem

If a language, L, is accepted by an NDT Turing machine, $\mathbf{T}$, then it is accepted by an NDO Turing machine $\mathbf{O}$ and vice versa.

The argument has too many words for a slide.

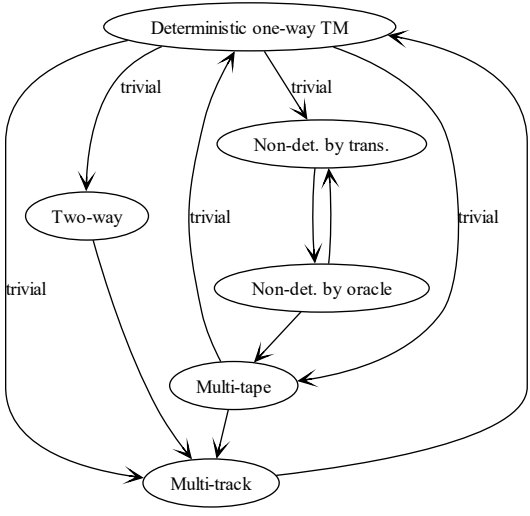
And it *still* doesn't matter

Theorem

If a language, L, is accepted by a non-deterministic TM then it is accepted by an ordinary TM.

- Simulate the possible computations of an NDO machine, O, using a multi-tape machine, T.
- T has: a working tape, a *clean input* tape (initialised to the actual input w that we want to check), a *simulated oracle* tape, and a *timer* tape (initialised to 1).
- For each value, k, of the timer, we simulate the operation of O on every possible string on the oracle tape of length $\leqslant k$, operating for k steps. Each such simulation starts by erasing the working tape, then replacing it by a copy of the clean input.
- We accept if any simulation ever accepts.
- If none of the simulations at timer value k accept, we increment the timer to $k + 1$ and start again.

Overview of relationships between TM variants



$A \to B$ means that all languages accepted by TM type A are also accepted by TM type B.