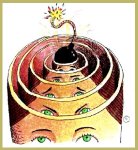
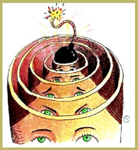


Lecture 39: Extensions to Turing Machines

Aims:

- To see that extensions to Turing machines add no power.



Module Home Page
Title Page
◀◀ ▶▶
◀ ▶

Back
Full Screen
Close
Quit

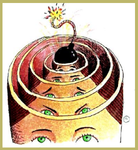
39.1. More Example Turing Machines

- By naming machines, we can combine complex machines into even more complex machines.
- For example,
  - Last lecture, we had a machine for copying its input. The copies are written with a space between them, and, when the machine halts, its read/write head will be over this space. Call this machine C.
  - The homework exercise was a machine that shifts a string left one cell. Call this machine S_L.
  - Now we can construct a machine which has the effect of creating copies without a space between them (by copying and then left-shifting the rightmost copy):

$$\begin{tikzcd}
> \mathrm{C} \arrow[r] & \mathrm{MR} \arrow[loop above, "\neq \square"] \arrow[r, "= \square"] & \mathrm{ML} \arrow[r] & \mathrm{SL}
\end{tikzcd}$$

  - In fact, in this example, the following also works:

$$C \longrightarrow S_L$$



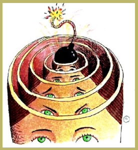
39.2. Extensions to Turing Machines

- The few examples that we've seen so far only hint at the power of Turing machines. But, it's natural to consider the effect of extending Turing machines: does this add any power?
- Many variations have been proposed. E.g.:
  - Allow the tape to have multiple-tracks.
  - Allow multiple read/write heads.
  - Allow many tapes, each with its own read/write head.
  - Allow a multi-dimensional tape.
  - Allow non-determinism.
  - Allow combinations of the above.
- They may bring greater convenience; they may speed up a computation. But none of them has increased the power. There are no problems that extended Turing machines can solve that standard Turing machines cannot solve.
- To show that a proposed extension is no more powerful than the standard model, we show

  anything the extension can do, the standard model can do.

  I.e. we would show how a standard Turing machine would simulate the operation of the extended machine.
- If we wanted to show they are of equal power, we would show

  anything the extension can do, the standard machine can do;
  anything the standard machine can do, the extension can do.

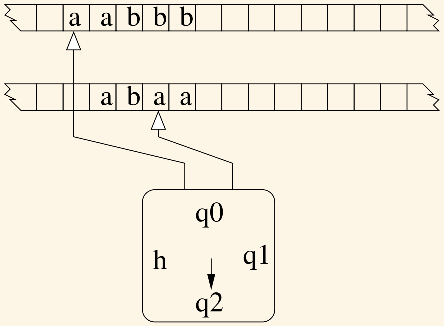


I.e. we would show how standard machines can simulate extended machines, and how extended machines can simulate standard machines.

- Here, we will only show the former: that the extension is no more powerful than the standard model.
- Furthermore, we won't give detailed proofs (because they're *very* detailed). We'll just sketch enough to get some intuitions.

39.2.1. Multiple Tapes

- In this extension, the machine can have several tapes, each with its own read/write head. We focus on the case where there are two tapes.



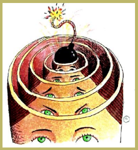
- In one step, the machine reads the symbols scanned by all heads and then, depending on those symbols and its current state, each head will move or write and the control unit will enter a new state. Note that the actions of the heads are independent of each other: in one step one head might move left and another might move right or write an a.

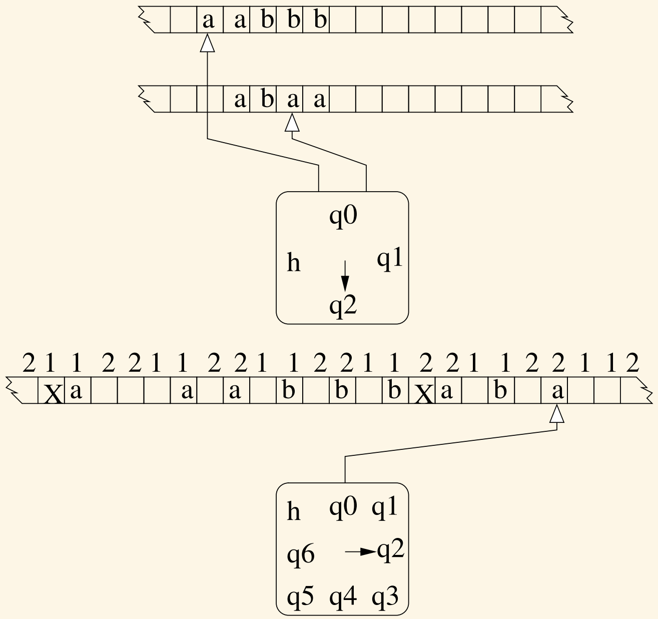
- The transition table might now look like this:

δ	a, a	$a, \sqcup$	$\sqcup, a$	$\sqcup, \sqcup$
q_0	R, L, q_0	$\sqcup, L, q_1$	$\sqcup, \sqcup, h$	R, R, q_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

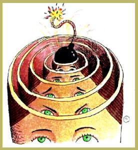
- This extension clearly brings
  - convenience — Imagine how easy it is to design a two-tape Turing machine that copies a string on one tape to the other tape; compare this with the (one-tape) Turing machine that copies its input from one part of its tape to another part of its tape.
  - efficiency — Similarly, consider the to-ing and fro-ing that can be saved by using two-tape machines.
- But it brings no extra power — as we shall now demonstrate.

- How can a Turing machine (i.e. a one-tape Turing machine) simulate a two-tape Turing machine?
- We interleave the contents of the two tapes onto a single tape.
  - First a symbol of tape 1, then the corresponding symbol from tape 2;
  - then the next symbol from tape 1 and the corresponding symbol from tape 2;
  - and so on.
- But we also leave an extra blank before each symbol. This can be used to record the position of the read/write heads.
- The control unit will also need a lot more states.





- The control unit of the one-tape machine will do a lot of work: whenever the two-tape machine does one step, the one-tape machine will do many steps.
- Whenever the two-tape machine does one step:
  - The one-tape machine scans left to see what symbols are next to the X's, i.e. to see what the two-tape machine would have read. Its control unit puts itself into a state that remembers this. E.g. there's one state to remember $\langle a, a\rangle$, another to remember $\langle a, \llcorner\!\lrcorner\rangle$, another to remember $\langle \llcorner\!\lrcorner, a\rangle$, and so on. Since Σ is finite,



there is a finite number of possibilities. (How many possibilities for a two-tape machine? How many for a k-tape machine?)

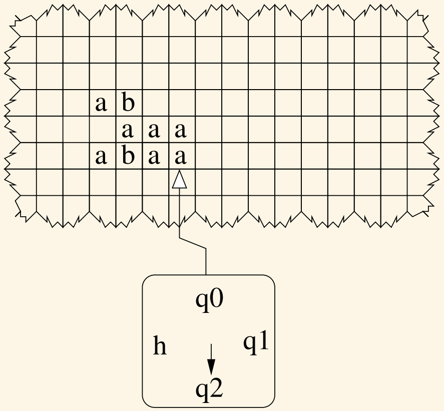
- It then scans right, back to the rightmost X.
- The control unit will now update the X's or the symbols to their right, in accordance with what the two-tape machine would have done.
  - E.g. if the two-tape machine would have instructed this head to write b, then go right from the X and write b.
  - E.g. if the two-tape machine would have instructed this head to move left, then write a blank, move 4 cells left (why 4?), and write an X.

It then finds the other X and repeats the above process.

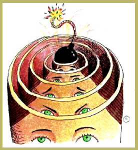
- It can be proven that this simulation correctly mimics the behaviour of the original two-tape machine.

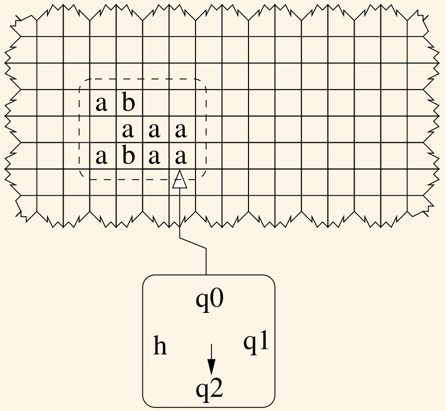
39.2.2. Multidimensional Tapes

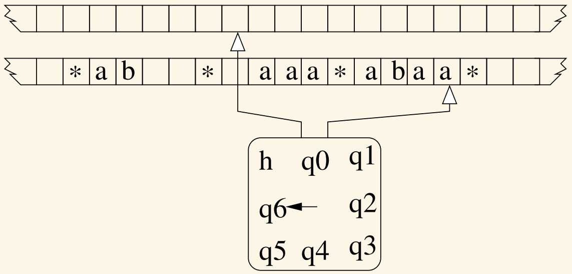
- In this extension, the machine has a single read/write head but a multidimensional tape. We focus on the two-dimensional case:



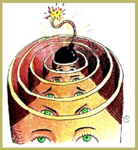
- In one step, the machine reads the symbol scanned by the head and then, depending on the symbol and its current state the control unit will enter a new state and the head will write a new symbol or move Right, Left, Up or Down.
- This extension brings no extra power — as we shall now demonstrate.
- How can a Turing machine (i.e. a Turing machine with a one-dimensional tape or tapes) simulate a Turing machine with a two-dimensional tape?
- We simulate using a two-tape Turing machine. But we already know that two-tape machines can, in their turn, be simulated by one-tape machines.
- We can draw a rectangle around the non-blank cells. Then copy each row of the rectangle onto the second tape, separated by, e.g., *.





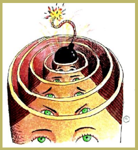


- Whenever the two-dimensional machine moves its head left or right, the two-tape machine moves tape 2's head left or right.
- Whenever the two-dimensional machine moves up (or down), the two-tape machine
  - scans left until it finds *;



  - as it scans, it writes symbols onto tape 1 to record how many cells it had to scan before it reached the *;
  - then it scans to the next * to the left (for up) or right (for down);
  - then it moves right a certain number of times, as recorded on tape 1.
  - If the result of all of this has taken it vertically out of the rectangle, then we add to tape 2 a new line of the same length to the left or right of the current lines.
  - If it has moved horizontally out of the rectangle, then we add one cell to the left or right of each line (which will require a lot of left or right shifting).

- It can be proven that this two-tape simulation correctly mimics the behaviour of the original two-dimensional machine.



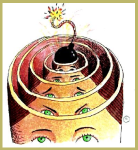
39.3. Concluding Remarks

- So extensions to Turing machines, e.g. multiple tapes or a multidimensional tape, add no power.
- In fact, these variations are of equivalent power to (standard) Turing machines.
- What about restrictions?
  - Some restrictions have no effect on the power.
    - E.g. A Turing machine that is required to leave the input intact, to clear up all its 'workings', so that, when it halts, the tape contains just the input and the output, surrounded by blanks.
    - E.g. A Turing machine that has no tape but instead has two stacks.
    - E.g. A Turing machine whose tape is infinite only to the right.
    - E.g. A Turing machine whose tape alphabet contains only two symbols.
    - E.g. A Turing machine whose control unit has only two states.
  - But, some restrictions do reduce the power.
    - E.g. A Turing machine that can traverse the tape in only one fixed direction. People call these *finite-state machines*. They don't usually define them in the way we have here (as restrictions of Turing machines). They usually come at them from a different starting point. But they amount to the same thing. They're quite useful in different parts of Computer Science. E.g. you might encounter them if, at some point, you study compilers.

Acknowledgements

This lecture owes a lot to [Jun] and a little to [LP81].

Clip Art (of head with bomb) licensed from the Clip Art Gallery on DiscoverySchool.com.



References

[Jun] A. Jung. Models of Computation (Course Notes).

[LP81] H.R. Lewis and C.H. Papadimitriou. *Elements of the Theory of Computation*. Prentice Hall, 1981.