CS411 Theory of Computation
Lecture 5

Sergey Kitaev

University of Strathclyde

4 October 2017

Multitape Turing Machines

Definition (Multitape Turing Machine)

- A multitape Turing Machine M is like an ordinary Turing machine. There are several (say k) tapes and every tape has its own head.
- The input is put on tape 1 and the other tapes are blank
- The transition function is altered to allow for reading, writing and moving the heads on tapes (sometimes simultaneously).
- Formally: $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, S\}^k$.
- The expression $\delta(q_i, a_1, \ldots, a_k) = (q_j, b_1, \ldots, b_k, L, R, \ldots, L)$ means: If M is in state q_i, and the tape heads currently read $(a_1, \ldots, a_k)$, then replace these with $(b_1, \ldots, b_k)$, move to state q_j, and move the heads left or right depending on the direction list.

Multitape Turing Machines

This multitape Turing machine sounds a lot fancier and more powerful that the ordinary Turing machine.

Question: Is it?

Answer: No.

We will now go about 'proving' this.

Theorem (Sipser 3.13)

Every multitape Turing machine has an equivalent single-tape Turing machine.

Proof of 3.13

This proof shows how to 'convert' a multitape Turing machine into an ordinary (single tape) Turing machine.

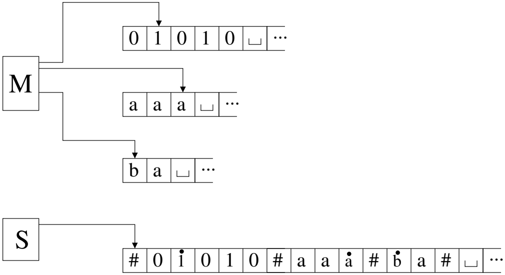
Let M be the multitape Turing machine that we are considering and suppose it has k tapes.

We will now construct an ordinary Turing machine S that simulates M.

Suppose that M has k tapes.

- S will simulate the effect of k tapes by storing their info on a single tape.
- It will use a new symbol # as a delimiter to separate the contents of the different tapes.
- The location of the head of each tape must be recorded and we do this by marking the symbols where the heads currently are with a dot.

The following diagram shows how to 'convert' a configuration on M into one on S:



Proof of 3.13 continued

$S =$ "On input $w = w_1 \cdots w_n$:

1. First S puts its tape into the format that represents all k tapes of M. The formatted tape contains:

$$\#\dot{w}_1 w_2 \cdots w_n \# \dot{\sqcup} \# \dot{\sqcup} \# \cdots \#$$

2.
   - To simulate a single move, S scans its tape from the first # (which marks the left end) to the $(k+1)$st # which marks the right end. It does this to determine the symbols under the virtual heads.
   - S makes a second pass to update the tapes according to the way that M's transition function dictates.
3.
   - If at any point S moves one of the virtual heads to the right onto a #, then this action signifies that M has moved the corresponding head onto the previously unread blank portion of that tape.
   - So S writes a blank symbol on this tape cell/position, and shifts the tape contents (from this cell until the rightmost #), one unit to the right.
   - Then it continues the simulation as before." □

A corollary

Corollary (Sipser 3.15)

A language is Turing-recognizable if and only if some multitape Turing machine recognizes it.

Proof.

- Suppose that a language A is Turing-recognizable.
- This means that some Turing machine, M say, recognizes it.
- The Turing machine M is a multitape Turing machine that has 1 tape.
- Therefore the language A is recognized by a multitape Turing machine.

Conversely,

- Suppose that M' is a multitape Turing machine that recognizes a language A. By Theorem 3.13, there is an equivalent ordinary Turing machine M'' that recognizes A. □

Non-deterministic Turing Machines

Definition

A non-deterministic Turing machine (NTM) is the same as an ordinary (deterministic) Turing machine, except that it allows for more than one possible action for a given (state,symbol) pair.

The transition function becomes a set of possible outcomes:

$$\delta : Q \times \Gamma \to \texttt{power set}(Q \times \Gamma \times \{L, R\})$$

The computation of a NTM is a tree whose possibilities correspond to different possibilities for the machine.

Every such path gives rise to a replica deterministic TM.

Non-deterministic Turing Machines: Input & Output

- Input is some word w, same as TM case.
- Output: There can be many different outcomes of what is on the tape. However, we still have a notion of accept and reject states.
- We say an NTM accepts the input w if there is some replica TM that accepts the input.
- We say the NTM rejects the input w if all replica TMs reject the input.
- Otherwise, the NTM does not halt.

Recognizable and decidable languages for NTMs

- A language A is said to be recognized by a NTM N if for every string w in A, the machine N accepts w.
- A language A is said to be decided by a NTM N if
  - N recognizes A **and**
  - if the input w is not in A, then N will reject the input w.

Theorem (3.16)

Every non-deterministic Turing machine has an equivalent deterministic Turing machine.

How to prove this:

- We will show that we can always simulate a NTM N with a TM D.
- Do this by making D try all possible branches of N's computation.
  If it encounters the accept state, then D accepts.
  Else D's simulation will not terminate.
- Traversing the tree of possibilities using breadth first search.
  (Otherwise an infinite branch might miss an accept state on a finite branch.)

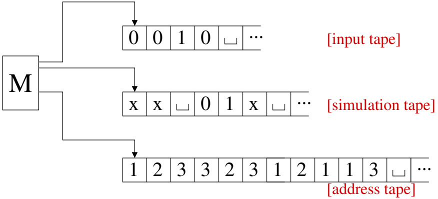
Proof of Theorem 3.16

The simulating deterministic TM D has three tapes.
Theorem 3.13 tells us this is equivalent to having just one tape.
D uses 3 tapes in the following way:

Tape 1: Contains input string.

Tape 2: Maintains a copy of N's tape on some branch of its non-deterministic computation.

Tape 3: Tracks of D's location in N's non-deterministic computation tree.



Proof of Theorem 3.16 contd

Associated with the running of N is the tree of N (on board).

Every node in the tree of N can have at most b children, where b is the largest set of choices in N's transition function.

Every node in the tree can be described by a string from the alphabet $\{1, \ldots, b\}$, where each successive value tells us which direction to branch in, starting from the root node.

For example, consider a transition function in which

$$\delta(q_1, 0) = \{(q_2, \text{x}, R),\ (q_1, 0, L),\ (q_3, 1, R)\}.$$

Proof of Theorem 3.16 contd

On input w:

1. Initially tape 1 contains the input w, and tapes 2 and 3 are empty.
2. Copy tape 1 to tape 2.
3. (i) Use tape 2 to simulate N with input w on one branch of its non-deterministic computation.
   (ii) Before each step of N consult the next symbol on tape 3 to determine which choice to make among those allowed by N's transition function.
   (iii) If no more symbols remain on tape 3 or if this nondeterministic choice is invalid, abort this branch by going to stage 4.
   (iv) Also go to stage 4 if a rejecting configuration is encountered.
   (v) If an accepting configuration is encountered, accept the input.
4. Replace the string on tape 3 with the lexicographically next string. Simulate the next branch of N's computation by going to stage 2. $\square$

Corollaries to Thm 3.16

Corollary (3.18)

A language is Turing-recognizable if and only if some NTM recognizes it.

Proof.

- If a language is Turing-recognizable, then some Turing machine recognizes it. Every deterministic TM is automatically a NTM. Therefore if a language is Turing recognizable, then some NTM recognizes it.
- Suppose an NTM N recognizes a language A. By Theorem 3.16 there is a Turing machine D that is equivalent to N, and which recognizes A.

$\square$

Corollaries to Thm 3.16

- The proof of Thm 3.16 can be modified so that if N always halts on all branches of its computation, then D will halt.
- We call a NTM a decider if all branches halt on all inputs.

Corollary (3.19)

A language is decidable if and only if some NTM decides it.