MTAT.07.004 — Complexity Theory	September 15th, 2011
Lecture 3	
Lecturer: Peeter Laud	*Scribe(s): Ilya Kuzovkin*

Introduction

During this lecture we went through proofs for four theorems.

Multi-tape Turing Machine → Two-tape Turing Machine

Theorem 1 *Let machine M with k tapes accept a language / compute a function in time T. There exists a TM M' with two tapes that accepts the same language / computes the same function in time $O(\lambda n.T(n)^2)$*

Intuitively put it means that if we can solve a problem on a turing machine with k-tapes we also can come up with a 2-tape turing machine which will be able to solve same problem in time $O(T(n)^2)$

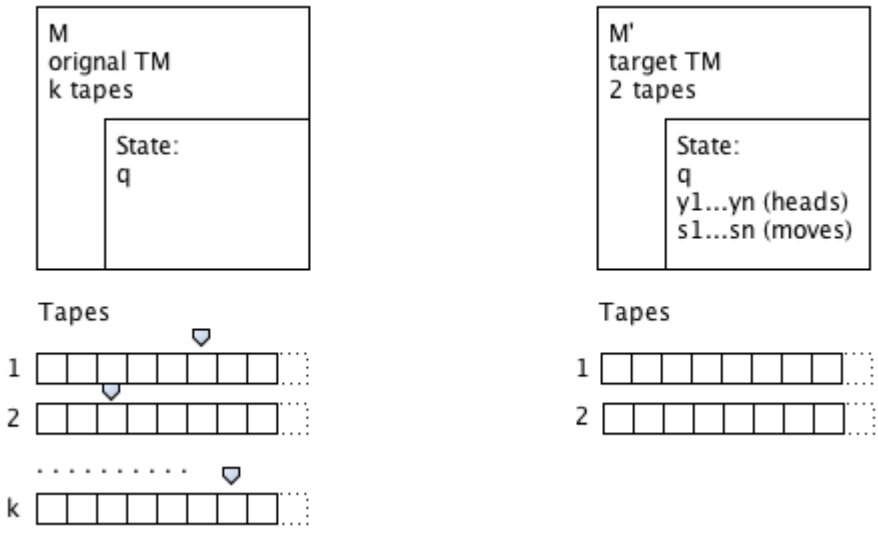

Figure 1: M and M'

Proof The original TM M had one input tape and $k-1$ work tapes. The machine M' we're constructing will have just a single working tape. To write all data from k tapes onto one tape we do the following: we take every first symbol from k tapes and put them onto k first positions of the target tape. First symbol from first tape goes to position 1, first

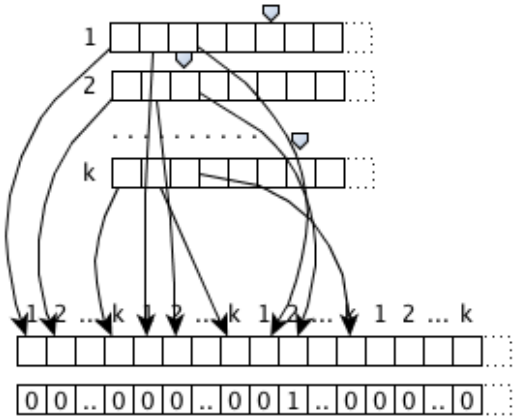

Figure 2: Representing k tapes on 2 tapes

symbol from second tape goes to position 2, first symbol of the kth tape goes to position k. Then we start transferring second symbols: second symbol from first tape goes to position k+1, second symbol from second tape to position k+2, kth symbol from second tape to position $k + k$ and so on. In such way we transfer all data from k tapes to one single tape.

Only thing we left to store is position of the heads. For recording those, we increase the tape alphabet. If Γ was the tape alphabet of M, then the tape alphabet of M' will be $\Gamma \times \{0, 1\}$. The second component indicates whether a head of some tape of M is located at this position (this is denoted as 1) or not (this is denoted as 0). As you can see on Figure 3, on the second tape we have head on position 3 – we mark corresponding position on the target tape as 1.

Performing one step on machine M takes constant time. You just pass the instructions to every head and it performs a step.

Now if we look at machine M' then we can see that for performing 1 step it has to scan through all the tapes to find out there heads are at the moment. After we have done that we will have to scan through the tape once more to apply all the operations we have to do to perform a step. So performing one step on machine M' will take time $2n \in O(n)$.

Consequently, if we want to perform n steps with temporal cost of $O(n)$ each, we will have to spend $O(n^2)$ time. ■

Theorem 2 *Let machine M with k tapes accept a language / compute a function in time T. There exists a TM M' with two tapes that accepts the same language / computes the same function in time $O(\lambda n.T(n)logT(n))$*

Proof To do that we have to come up with more sophisticated way of representing k tapes on 3 tapes. In this machine we will use 3 tapes instead of 2. Third tape will be used to temporary store data we want to move during the step.

We imagine our resulting data tape in machine M' has k layers – one for each tape in machine M.

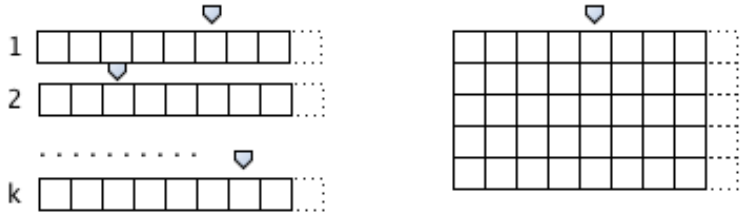

Figure 3: Representing k tape as one tape with k layers

We have only one head in machine M'. To simulate movement of the head in M we move the corresponding layer of the tape of M'. For example if we move second head of the original machine to the left, we move second layer to the right. Such operation will require to rewrite every symbol on the layer we moved, which will take linear time. So this "layer"-trick alone will do no good – we still will have to do n operations in M' for every step in M.

Next trick we make we split each layer into groups R_i and L_i in such way that $|R_1| = |L_1| = 1$ and $|R_i| = |L_i| = 2|R_{i-1}| = 2|L_{i-1}|$

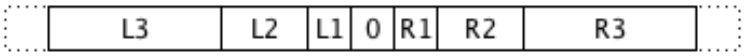

Figure 4: Splitting layer into groups of exponentially growing size

We introduce new symbol $\emptyset$ to the alphabet, which will denote empty space. For the machine M', the symbol $\emptyset$ will serve as the blank symbol on the tapes. The blank symbol of M is just an ordinary symbol for M'. We prescribe the following invariants for the contents of the cells in R_i and L_i

- Each L_i, R_i is one of $\{$*empty*, *full*, *half-empty* = *half-full*$\}$
  - Here "empty" means that all cells in the group are $\emptyset$. "Full" means that none of the cells are $\emptyset$. "Half-full" means that *exactly* half of the cells are $\emptyset$.
- $R_i + L_i$ is always *half-empty*
- Position 0 is always *full*, so we can always read from it

Consider we want to make one step to the left. Then we will move the layer to the right. If L_1 is *empty* then there is no problem, we move data from 0-position to L_1, from R_1 to 0-position and we are done.

But it may happen that L_i is *full* and we cannot put anything there. In that case we act as follows:

1. Move head to the right until *not empty* cell R_i found ($i = min(R_i$ not empty$)$). Suppose in our example $i = 3$, which means R_3 is first *not empty* cell

2. Write elements from R_3 to the third tape

3. Move these element to the cells R_2 and R_1 making them *half-full*

4. In the mirrored way cells L_1 and L_2 will be moved to cell L_3 to preserve the balance rule "$R_i + L_i$ are always *half-empty*". So if L_1 and L_2 were *full* they will become *half-full*. L_i will become *half-full* (if it was *empty*) or *full* (if it was *half-full*)

5. In the end of those transactions cells $\{L_{i-1}, \dots, L_2, R_2, \dots R_{i-1}\}$ are *half-full*

This corresponds to 1 step in original machine M. Time in M' will be $O(2^i)$, where i goes up to log(Time of orignal machine M). Pair $R_i \& L_i$ is considered at most once per 2^{i-2} steps. Knowing number of time they are considered and time cost of one step we can calculate total running time of machine M'

$$T' \leq$$

$$\leq \sum_{i=1}^{\lceil \log T \rceil} (\text{number of times } R_i \& L_i \text{ are considered}) \cdot (\text{time cost of the step}) \leq$$

$$\leq \sum_{i=1}^{\lceil \log T \rceil} \frac{T}{2^{i-2}} \cdot O(2^i) \leq^{\text{see Note}}$$

$$\leq \sum_{i=1}^{\lceil \log T \rceil} T \cdot O(1) =$$

$$= \log T \cdot T \cdot c$$

Note This step can be done because $\frac{O(2^i)}{2^{i-2}} = c$ (constant)

$$logT \cdot T \cdot c \in O(T \log T)$$

■

Speeding up a TM

Theorem 3 *Let a TM M compute a function / accept a language in time T. Then for each $c \in \mathbb{N}$ there exists a TM M' and constant c' such that M' computes the same function / accepts the same language in time $\lambda n. \frac{1}{c} T(n) + c'n$*

In other words we can speed up a Turing Machine by any number of times c, but we receive penalty from c' (at some point penalty will be so big that there will be no point in increasing c).

Proof

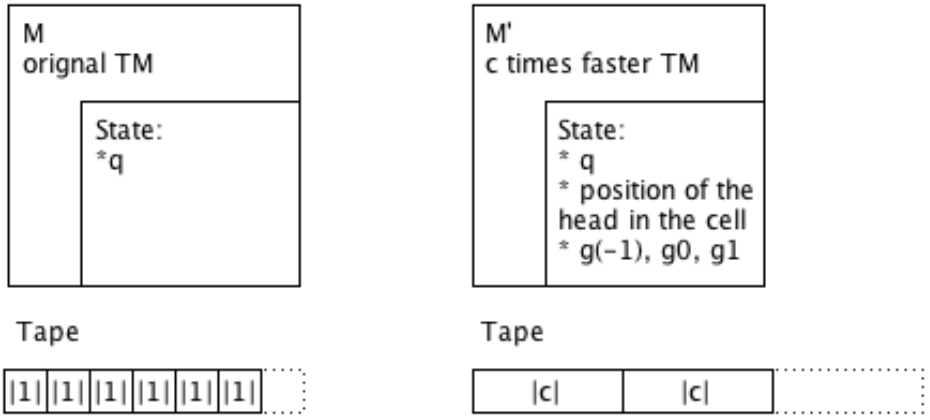

Figure 5: Original machine M and machine M' which works $\approx c$ times faster

Assume $c = 5$ and we have a piece of tape of length 100.

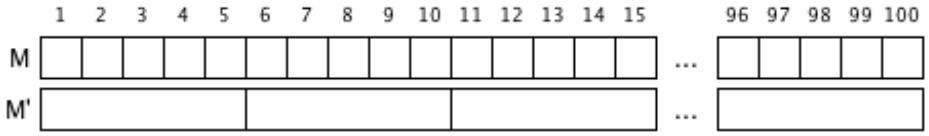

Figure 6: Two tapes of two different machines which hold same 101 symbols

Now imagine a task "read 99th symbol on the right". On the M it will look like "*go 99 times to the right, read*", which is 100 steps. On the M' same task will look like "*go 19 times to the right, read*" (note that during the read, the contents of all five cells of the tape of the original TM will be read), which is $19 + 1 = 20$ steps. The speedup will be $\frac{100}{20} = 5$ and you can see with $n \to \infty$ this number will go to c. Penalty $c'n$ comes from the requirement to encode the input of the TM M.

■

SAT problem is NP-complete

Theorem 4 *SAT is NP-complete.*

Proof Assume we have NTM which solves some particular problem in polynomial time. This NTM defines its *configuration graph* where the nodes are the configurations of this NTM and there is an arc from one configuration to another one if the NTM can go from the first configuration to the second one in a single step. Notice that because of the non-determinism, there can be several outgoing arcs in each node.

A configuration $\langle q, tapes, heads \rangle$ of the NTM is

- q is current state
- *tapes* is set of words $w_1, \ldots, w_k$ on all k tapes
- *heads* is a set of positions of the heads $s_1, \ldots, s_k$

Let $Q = \{q_1, \ldots, q_N\}$ be the set of states of our NTM and $\Gamma = \{\gamma_1, \ldots, \gamma_m\}$ be its tape alphabet.

Let x be the input to the NTM. The NTM accepts x iff there exists a path of length polynomial in $|x|$ from the initial state corresponding to x to some accepting state. As the length of the path must be polynomial, the configurations it passes are of polynomial size as well. Let ℓ be the maximum size of configurations we have to consider.

We can encode a configuration with the help of the following boolean variables.

- The variable state$_i$, where $1 \leq i \leq N$. The variable $state_i$ is true the current state in the configuration is q_i
- The variable Tape[r, s, t] denotes the fact that on the tape r in position s there is symbol t
  - Length of word on the tape $|w_i| \leq \ell$
  - $1 \leq r \leq k$
  - $1 \leq s \leq \ell$
  - $1 \leq t \leq m$
- The variable $pos[r, h]$ denotes the fact that head of the tape r is on position h

NTM can be represented as a graph

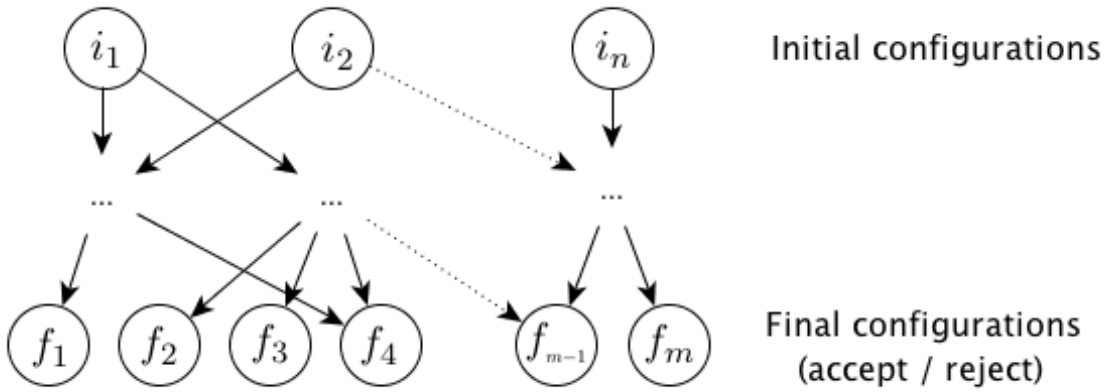

Figure 7: NTM as a graph

Now we represent the graph as Boolean formula:

- Each vertex is a NTM configuration C_i and in formula it is encoded through a number of boolean variables $u_1^i, \ldots, u_n^i$
- The formula $S(u_1, \ldots, u_n)$ indicates that the values of the variables $u_1, \ldots, u_n$ encode an actual configuration that is present in the graph.
- The formula $R(u_1, \ldots, u_n, u'_1, \ldots, u'_n)$ indicates whether we can go from configuration C encoded by $u_1, \ldots, u_n$ to the configuration C' encoded by $u'_1, \ldots, u'_n$ in a single step.
- The formula $\Phi^\circ(u_1, \ldots, u_n)$ indicates that $u_1, \ldots, u_n$ encode the initial configuration of the NTM corresponding to the input x.
- The formula $\Phi^\bullet(u_1, \ldots, u_n)$ indicates that $u_1, \ldots, u_n$ encode an accepting final configuration.
- Path from the initial configuration corresponding to x to a final configuration is the conjunction of all necessary conditions for the path to exist:
  1. $\Phi^\circ(u_1^0, \ldots, u_n^0)$ exists
  2. Each intermediate configuration exists: $S(u_1^i, \ldots, u_n^i)$
  3. There are all necessary transitions from C_i to C_{i+1}
  4. At least one of the configurations we will end up in is final configuration: $\bigvee_{i=0}^{k} \Phi^\bullet(u_1^i, \ldots, u_n^i)$

  All these conditions can be expressed as one Boolean formula, if it is *true* then path exists.

$$\Phi^\circ(u_1^0, \ldots, u_n^0) \wedge \left(\bigwedge_{i=1}^{\ell} S(u_1^i, \ldots, u_n^i) \wedge R(u_1^{i-1}, \ldots, u_n^{i-1}, u_1^i, \ldots, u_n^i) \right) \wedge \left(\bigvee_{i=0}^{\ell} \Phi^\bullet(u_1^i, \ldots, u_n^i) \right)$$

where S is

$$S \equiv \text{exactly\_one}(state_1, \ldots, state_N) \wedge$$
$$\bigwedge_{r,s} \text{exactly\_one}(tape[r, s, 1], \ldots, tape[r, s, m]) \wedge$$
$$\bigwedge_{r} \text{exactly\_one}(pos[r, 1], \ldots, pos[r, \ell])$$

which means that machine can be exactly in one state, on every tape there can be exactly one symbol on each position and every tape's head can be exactly in one position at a time. exactly_one can be formally expressed as

$$\text{exactly\_one}(x_1, \ldots, x_n) = \neg(\neg x_1 \wedge \ldots \wedge \neg x_n) \wedge \bigwedge_{1 \le i \le j \le n} (x_i \Rightarrow \neg x_j)$$

and R is

$$R \equiv \bigvee_{(q, \gamma_1, \ldots, \gamma_k, q', \gamma'_1, \ldots, \gamma'_k, p_1, \ldots, p_k) \in \sigma} \Bigg(\text{// find one transition which can be done}$$

$$state_q \wedge \left(\bigwedge_{r,s} pos[r,s] \Rightarrow tape[r,s,\gamma_r] \right) \wedge \quad \text{// state before transition}$$

$$state'_{q'} \wedge \left(\bigwedge_{r,s} pos[r,s] \Rightarrow tape'[r,s,\gamma'_r] \right) \wedge \quad \text{// state after transition}$$

$$\left(\bigwedge_{r,s} pos[r,s] \Rightarrow pos[r,s+p_r] \right) \wedge \quad \text{// heads change position}$$

$$\left(\bigwedge_{r,s} \neg pos[r,s] \Rightarrow \bigwedge_{t} tape[r,s,t] \Leftrightarrow tape'[r,s,t] \right) \right) \quad \text{// symbols not under the heads do not change}$$

The transition σ is defined as $\sigma \subseteq (Q \times \Gamma^k) \times (Q \times \Gamma^{k-1} \times \text{Move}^k)$

We put R and S into the Boolean formula and if it is satisfiable then is satisfying valuations encode possible paths from the initial state to accepting final states, which means that NTM accepts.

■