Advanced Complexity **(CSE 201a, Spring 2011)**

Lecture 4: Simulations of Turing Machines

April 12

Lecturer: Russell Impagliazzo *Scribe: Daniel Moeller*

1 Today's Plan

- Hierarchy Theorems
- Oblivious TMs
- Simulations of TMs by circuits, other TMs
- Reductions to 3-SAT for NE

2 Time Hierarchy Theorems

The following two theorems serve as a review of diagonalization techniques. The first uses a more basic technique, while the second requires a more sophisticated diagonalization argument.

Theorem 2.1. *$DTIME(t(n)) \neq DTIME(t'(n))$ for $t'(n) << t(n), t(n)$ time constructible*

Proof. Choose $t''(n)$ such that $t'(n) < t''(n) < t(n)$ (i.e. $\sqrt{t'(n)t(n)}$). Define a language

$$L_{t''} = \{(M, x) | M \text{ does not accept } (M, x) \text{ in at most } t''(n) \text{ steps}\}$$

Clearly $L_{t''} \in \text{DTIME}(t(n))$ since $t(n) > t''(n)$ so we can simply run $M(M, x)$ for up to $t''(n)$ steps and report whether it accepts or not. Now consider any machine M_0 which accepts $L_{t''}$. Suppose M_0 accepts $L_{t''}$ in at most $ct'(n)$ steps for some constant c. Then choose x so that $ct'(n) < t''(n)$. We have:

$$\begin{aligned}
(M_0, x) \in L_{t''} &\Leftrightarrow M_0 \text{ does not accept } (M_0, x) \text{ in } t''(n) \text{steps (by definition of } L_{t''})\\
&\Leftrightarrow (M_0, x) \notin L_{M_0} \text{ (since } M_0(M_0, x) \text{ would have to accept in } ct'(n) < t''(n) \text{ steps)}\\
&\Leftrightarrow (M_0, x) \notin L_{t''} \text{ (since } M_0 \text{ accepts all } (M, x) \in L_{t''})
\end{aligned}$$

This is a contradiction so M_0 cannot accept $L_{t''}$ in at most $ct'(n)$ steps. Therefore $L_{t''} \notin \text{DTIME}(t'(n))$ so $\text{DTIME}(t(n)) \neq \text{DTIME}(t'(n))$. □

Note that the above proof works if $t(n) > t''(n)\lg(t''(n))$ so that $t(n) \geq \omega(t'(n)\lg(t'(n)))$.

Theorem 2.2 (Fortnow and Santhanam)**.** *NTIME$(t(n)) \neq$ NTIME$(t'(n))$ for $t'(n) << t(n), t(n)$ time constructible*

Proof. Again, choose $t''(n)$ such that $t'(n) < t''(n) < t(n)$. Then let

$$\Lambda_{t''} = \left\{ (M, x, y) \middle| \begin{array}{ll} \text{If} & |y| < t''(n), M \text{ accepts both } (M, x, y0) \text{ and } (M, x, y1) \text{ in at most } t''(n) \text{ steps.} \\ \text{If} & |y| = t''(n), M \text{ rejects } (M, x, \epsilon) \text{ on a path determined by } y \end{array} \right\}$$

where we can think of M as a $t''(n)$-time machine and y as a partial description of $t''(n)$ bits of nondeterminism M uses. Then $\Lambda_{t''} \in$ NTIME$(t(n))$ if we consider $t(n)$ to be the time to simulate $t''(n)$ steps. Again, consider any machine M_0 which accepts in at most $ct'(n)$ steps for some constant c. Then choose x so that $ct'(n) < t''(n)$.

$$\begin{aligned} (M_0, x, \epsilon) \in L_{M_0} &\Leftrightarrow (M_0, x, 1) \in L_{M_0} \text{ and } (M_0, x, 0) \in L_{M_0} \text{ (by definition of } \Lambda_{t''}) \\ &\Leftrightarrow (M_0, x, 11) \in L_{M_0}, (M_0, x, 10) \in L_{M_0}, (M_0, x, 01) \in L_{M_0}, \text{ and } (M_0, x, 0) \in L_{M_0} \\ &\vdots \\ &\Leftrightarrow (M_0, x, y) \in L_{M_0} \forall y \in \{0,1\}^{t''(n)} \text{ (by definition of } \Lambda_{t''}) \\ &\Leftrightarrow (M_0, x, \epsilon) \text{ rejects on path } y \forall y \in \{0,1\}^{t''(n)} \\ &\Leftrightarrow M_0 \text{ rejects } (M_0, x, \epsilon) \end{aligned}$$

which is a contradiction. Therefore there is no machine $M_0 \in$ NTIME$(t'(n))$ which can accept $\Lambda_{t''}$ so NTIME$(t(n)) \neq$ NTIME$(t'(n))$. □

3 Oblivious TMs

In order to simulate a TM with a circuit, it is first useful to consider the following form of TM.

Definition 3.1. An *Oblivious Turing Machine* (OTM) is a machine for which, at every time t, no matter what input we have, the machine head is at cell $s(t)$ for some function s.

A naive way to simulate any TM with an OTM is to mark where the head of the tape is and then scan the tape to find the marker at each step. However, this method will produce a $O(T^2)$-time OTM for a T-time TM. The following theorem provides a much better simulation.

Theorem 3.2 (Pippenger)**.** *Any T-time Turing Machine can be simulated by an $O(T \lg T)$ time Oblivious Turing Machine. Moreover, all but one tape share the same computable $s(t)$.*

Proof. Consider the following Oblivious Simulation algorithm which simulates a TM

[H] **OS**(2^i) $i < 1$ Simulate in quadratic time. each tape Break the first $8 * 2^i$ cells of each tape into 8 strips of size 2^i
Obliviously swap the 4 cells closest to the tape head with the 4 at the front
this can be done obliviously by scanning the set of cells 8 times and placing each cell in the appropriate place on a secondary tape. **OS**(2^{i-1})
OS(2^{i-1})
each tape Obliviously swap everything back to the original position.

The intuition behind this algorithm is that in 2^i time the tape head will not go too far from its current position. Therefore we want the "nearby" tape squares to be at the front of the tape so we can know where the head is at all times.

Now the running time of this algorithm follows the recursion $T(2^i) = O(2^i) + 2T(2^{i-1})$ so, in general, $T(N) = O(N) + 2T(\frac{N}{2}) = O(N \lg N)$.

Moreover, since we are performing the same computation on all tapes except the secondary tape used for oblivious swapping, all but that tape share the same $s(t)$. □

Note that any such $k+1$-tape OTM that takes time T can be simulated by an O(kT)-time 2-tape OTM by interleaving the k tapes that share the same $s(t)$. Thus, for fixed k this is still $O(T)$.

4 Simulating TMs by Circuits

A naive simulation of a TM by a circuit would use constantly many gates to check that each cell position is correct at every time step, by checking that the neighboring cells in the current time step and the previous time step are valid. However, this results in an $O(T^2)$-size circuit simulation of a T-time TM since there are $O(T)$ cell positions to check at each time step. However, using an OTM, we can obtain the following result.

Theorem 4.1. *An Oblivious Turing Machine can be simulated by a circuit with constant overhead.*

Proof. Now we can make use of the function $s(t)$ to tell us where the tape will be at any given time. Therefore the gates can follow the path of the OTM and we only need to check if it was correct based on the last time the location was visited.

There are two things that must be verified:

1. What is the state of the machine at time t?
2. What is written on cell $s(t)$ at time t?

1. depends only on cell $s(t-1)$ at time $t-1$ and the state at time $t-1$, while 2. depends on the state at time t and what was written on $s(t)$ at the most recent time t' such that $s(t') = s(t)$. Again, each of these verifications requires only constantly many gates. Therefore since we only need to verify a constant number of cells at each time, the overhead is constant. □

The following corollary follows directly from the previous two theorems.

Corollary 4.2. *Any T-time Turing Machine can be simulated by an $O(T \lg T)$-size circuit.*

5 Reductions to 3-SAT for NE

Finally let us consider reductions to 3-SAT. In this case, we can reduce each cell of six tape squares formed by taking the cell and its two neighbors at the current and previous time steps to variables. This will produce $O(T^2)$ variables and T^{k+1} clauses, each involving $O(k|\Sigma||Q|)$ boolean variables, where k is the number of tapes, Σ is the alphabet, and Q is the set of states. This can further be reduced to a 3-SAT problem.

However, using an OTM, we only need to look at the path that follows the heads on the OTM, given by $s(t)$. Therefore we only need to look $O(T)$ variables and $O(T)$ clauses for fixed k. Nevertheless, we still will have $O(k|\Sigma||Q|)$ boolean variables per clause.

Using Theorem 3.2, it will take $O(T \lg T)$ for a non-oblivious TM.