CSE 200 Lecture Notes
Single vs. Multitape Turing Machines

Chris Calabro

September 29, 2006

1 Simulation time upper bound

It is possible to simulate t steps of a 2-tape TM M by a 1-tape TM in $2t^2 + O(t)$ steps. The idea is to expand the alphabet from Σ to an alphabet Σ' of pairs of symbols from Σ, each of which may possibly have a dot over it. The ith symbol in the pair at cell j will represent the symbol on the ith tape of M at cell j, and will have a dot over it iff the tape head of M is at cell j. Formally, $\Sigma' = (\{0, 1\} \times \Sigma)^2$, which has size $4|\Sigma|^2$.

To simulate 1 step of M, the new machine M' will scan the entire input and remember the 2 symbols with the dots over them in its finite memory. This takes time $\leq t$. Then M' will scan back and update the 3 symbols around each of the dotted symbols according to the transition function of M. This takes time $\leq t + O(1)$. So the total time to simulate t steps is $2t^2 + O(t)$.

2 Simulation time lower bound

We claim that to simulate a 2-tape TM for t steps by a 1-tape TM may require $\Omega(t^2)$ steps. Let

$$L_n = \{x0^n x \mid x \in \{0, 1\}^n\}$$
$$L = \cup_{n \in \mathbb{N}} L_n.$$

That L can be decided in $O(n)$ time by a 2-tape TM is left as an exercise. But we claim that every 1-tape TM M that decides L runs in time $\Omega(n^2)$. The idea is that with its finite number of states, M can only remember $O(1)$ bits at a time and must ferry $O(n)$ bits of information back and forth across the 0^n region of the input, and so must use $\Omega(n^2)$ time (each trip carries $O(1)$ bits and so $\Omega(n)$ trips are required, and each trip takes $\Omega(n)$ time.) But proving this idea rigorously will require a serious trick. [1]

[1] proof adapted from lecture notes of EECS 574, Fall 2002, at the University of Michigan; original proof by J. Hartmanis, 1968

Define the (w, i) *crossing sequence* $C(w, i)$ for M as the sequence of states that M has when its head crosses from cell i to $i+1$ or from $i+1$ to i (counting the first cell as cell 0), in chronological order. Then the time M takes on input w is

$$T(w) = \sum_i |C(w, i)|. \tag{1}$$

Let

$$T(n) = \max_{w \in \{0,1\}^{3n}} T(w).$$

Notice also that if M accepts strings x_1x_2 and y_1y_2, and $k = |x_1| = |y_1|$, and $C(x_1x_2, k) = C(y_1y_2, k)$, then M accepts both x_1y_2 and y_1x_2, since after crossing the k to $k+1$ boundary, M cannot remember which string it was looking at.

We claim that

$$\exists i_0 \in I \left| \left\{ w \in L_n \mid |C(w, i_0)| \le \frac{T(n)}{n-1} \right\} \right| \ge \frac{2^n}{n}, \tag{2}$$

where $I = [n, 2n)$. For suppose not. Then

$$\begin{aligned}
2^n T(n) &\ge \sum_{w \in L_n} \sum_{i \in I} |C(w, i)| \quad \text{from (1)} \\
&= \sum_{i \in I} \sum_{w \in L_n} |C(w, i)| \\
&> n\Big(1 - \frac{1}{n}\Big) 2^n \frac{T(n)}{n-1} \quad \text{from the negation of (2)} \\
&= 2^n T(n),
\end{aligned}$$

a contradiction. So let $W = \{w \in L_n \mid |C(w, i_0)| \le \frac{T(n)}{n-1}\}$.

Define the map f with domain W by $f(w) = C(w, i_0)$. We claim that f is injective. For if not, then $\exists x_1x_2, y_1y_2 \in W, |x_1| = |y_1| = i_0, x_1x_2 \ne y_1y_2$ with $C(x_1x_2, i_0) = C(y_1y_2, i_0)$. But then $x_1y_2 \in L_n$, a contradiction. So, letting s be the size of the state set of M, we have

$$\begin{aligned}
\frac{2^n}{n} &\le |W| = |\operatorname{ran} f| \\
&\le \sum_{j=0}^{\frac{T(n)}{n-1}} s^j \quad \Big(\text{since each } f(w) \text{ has size} \le \frac{T(n)}{n-1}\Big) \\
&\le s^{\frac{T(n)}{n-1}+1}.
\end{aligned}$$

Since s is a constant, taking logarithms of both sides shows that $T(n) \ge \Omega(n^2)$.