Assignment #1
Advanced Theory of Computation (COMP 531)

Guillaume Labranche (260585371)

due on 27 January 2016

1. Starting with a TM A space-bounded $T(n) = O(S(n))$ which recognizes L, we define a new TM B space-bounded $S(n)$ which will simulate A and detect loops in its linear configuration history using only a constant amount of extra memory.

   The trick for loop detection with constant memory is to store two computation configurations, moving one forward by 1 step and the other by 2 steps using A's transition function. Eventually if there is a loop, both configurati$b_{i-1} + 1$ons will be equal.

   Lemma 1. *Two pointers a and b moving forward by 1 and 2 nodes respectively every step in a possibly infinite linked list will eventually be equal* ***iff*** *there is a loop in the list.*

   Proof. ($\Longrightarrow$) Since b is ahead of a, the only way for a to catch up to b is if b follows a node that points to another node before a. b will then catch up to a. In that case we clearly have a loop.

   ($\Longleftarrow$) If there is no loop, it is clear that a will never catch up to b and they won't be equal. If there is a loop, eventually both a and b will both end up in that loop, traversing it forever. Because a and b advance by 1 and 2 nodes, the distance from b to a gets smaller by 1 at every step. Therefore since the loop has a finite size, b will eventually catch up to a. $\square$

   Since the amount of possible configurations of A is $c^{S(n)}$, when A does not halt, by the pigeonhole principle it must end up in the same state at least twice. It is also true that if a TM is in the same state twice, it will not halt as its behaviour is dependant solely on its configuration. Therefore when B detects a loop, it rejects because we know that A does not halt on words not in L.

2. We describe a single-tape TM M who recognizes the language $L = \{a^n b^n : n \geq 0\}$ in time $O(n \log n)$.

   The trick is to keep a counter for n which we shift through our pass of the tape, incrementing while we process the a's and decrementing through the b's. At the end, we only need to check that the counter is 0. The shifting, incrementing and decrementing operations all take $O(\log n)$ steps. At every step of our pass (of which there are n), we shift and increase/decrease the counter. Our result time complexity is then $O(n \log n)$.

   In order to show that $O(n \log n)$ is the best possible, we rely on the proven theorem (by Hartmanis, Lewis & Stearns) that any single-tape machine running in time $o(n \log n)$ can only recognize regular languages and the fact that L is clearly not regular.

3. *Proof.* (by contradiction)

   Suppose there is such a non-regular language L recognized by a TM M in space $S(n) = o(\log\log(n))$. Thus $\forall k$, $\exists$ input w that requires more than k cells.

   Let w be the smallest such input and let $n = |w| > k$. Let C be the set of all possible configurations of the machine. A configuration depends on the current state, the tape content and the position of the tape head. The number of possible configurations is:

   $$|C| = |Q| \cdot S(n) \cdot |\Gamma|^{S(n)} \leq c^{S(n)} \text{ for some constant } c$$

   Let $C_i(x)$ denote the sequence of configurations at boundary i while reading x (called crossing sequence). Whenever M crosses the boundary i (located between cells i and $i+1$), the current configuration is appended to $C_i(x)$.

Let S denote the set of all possible crossing sequences while reading w. The maximum length of a crossing sequence is $|C|$ since otherwise M would end up in the same state twice and loop forever. Any element in a crossing sequence is a choice of $|C|$ possible configurations by definition. Thus the size of S is:

$$|S| = |C|^{|C|} = (c^{S(n)})^{c^{S(n)}} = c^{S(n) \cdot c^{S(n)}} < c^{c^{c \cdot S(n)}} = c^{c^{c \cdot o(\log \log n)}} = o(n)$$

Therefore the number of possible crossing sequences is less than a constant factor of the length of w. By the pigeonhole principle, this means that $\exists i, j, i < j$ such that $C_i(w) = C_j(w)$. If two different boundaries have the same crossing sequence, then removing the cells between such boundaries produces a new input w' that leads to the same result and uses as many cells as w. So $|w'| < |w|$ yet we defined w to be the smallest input in L that uses more than k cells. Therefore such w cannot exist for all k and L must be regular.

□

4. Let L be the language defined informally as the sequence of natural numbers in binary form. More formally:

$$L = \{b_0 \# b_1 \# \cdots \# b_n : n \geq 1, b_i \in \{0,1\}^* \text{ is the binary representation of } i\}$$

Proposition 1. *L is recognizable in $DSPACE(\log \log n)$.*

Proof. When parsing b_i, we need to make sure that $b_i = b_{i-1} + 1$. This can be done by first increasing b_{i-1} by 1 (constant memory needed for the remainder) and comparing it to b_i. We can compare two numbers one digit at a time. This requires 1 memory for the digit value and $\log \log i$ for its position in b_i since the size of b_i is $\log i$. We do so for every pair of b_i and b_{i+1}. Since the largest number we verify is b_n of size $\log(n)$, we need only $\log \log n$. □

Proposition 2. *L is not regular.*

Proof. L cannot be regular since in order to verify that a b_i is indeed the successor of b_{i-1}, we must verify that both numbers match up. This comes down to recognizing the language $\{ww : w \in \Sigma^*\}$ which is known to not be regular. □

5. We need this assumption in order to modify the encoding w without changing the behaviour of M_w. Because we are feeding w into M_w, we are essentially able to modify $|w|$ and $f(|w|)$.

   At the end of the proof, we suppose that we have M time-bounded by $f(n)$ recognizing K. We then take a long enough encoding w of M. We are able to draw a contradiction:

   - if M accepts w (i.e. itself) within $f(|w|)$ steps, then this means $w \in K$ but we defined K as the set of machines which do not accept within $f(|w|)$ steps.
   - if M does not accept w within $f(|w|)$ steps, then this would mean that $w \notin K$ whereas K as defined would contain w.

   Therefore K is not recognizable in time $f(n)$ but is computable, for any (space-constructible f).