Small Space = No Space

Chris Calabro

July 10, 2007

Abstract

One might think that having a super constant amount of space that is much less than \( \lg \lg n \) is still more powerful than having no space at all, but this is not the case. We show that

\[
\text{SPACE}(lglglgn) = \text{SPACE}(O(1)) = \text{SPACE}(0) = \text{the regular languages}.
\]

First we show

\[
\text{SPACE}(\lg \lg \lg n) \subseteq \text{SPACE}(O(1)).
\]

Let \( M \) be a TM using space \( \leq \lg \lg \lg n \) and tape alphabet \( \Gamma \), and let \( L \) be its language. Let \( f(n) = \max_{x \in L \cap \Sigma^n} \text{space } M(x) \). If \( f \leq O(1) \), then we are done. (Actually this isn’t so obvious since we defined \( f \) in terms of strings that \( M \) accepts, but if \( M \) uses \( c = O(1) \) space on the strings in \( L \), then we can construct a machine \( M' \) that uses \( O(1) \) space on all strings by simulating \( M \) and rejecting if it tries to use more than \( c \) space.)

Else \( \forall n_0 \exists n \geq n_0 \forall m < n \ f(m) < f(n) \). I.e. for infinitely many \( n \), \( f(n) \) strictly dominates \( f(m) \) when \( m < n \). For an \( n_0 \) to be chosen later, choose \( n \geq n_0 \) so that \( \forall m < n \ f(m) < f(m) \). Let \( s = f(n) \) and choose \( x \in L \cap \Sigma^n \) so that \( M(x) \) uses space exactly \( s \).

Define the total state of \( M \) when its input tape head is in position \( i \) as the state and the contents of the work tape. Define a crossing sequence at \( i \) as the sequence of total states when the input tape head of \( M \) crosses from position \( i \) to \( i+1 \) or vice versa. Notice that no crossing sequence can have length \( > |Q||\Gamma|^s \) because otherwise \( M \) would loop, contradicting that \( M \) accepts \( x \).

So the number or crossing sequences at \( i \) is

\[
\leq (|Q||\Gamma|^s)^{|Q||\Gamma|^s+1} \leq 2^{2^{O(lg lg lg n)}} = 2^{(lg lg n)^O(1)} \ll n.
\]

We retroactively choose \( n_0 \) large enough so that the above inequality holds \( \forall n \geq n_0 \).

So \( \exists 1 \leq i < j \leq n \) s.t. the crossing sequence at \( i \) is the same as that at \( j \). Let \( y = x \) but with the substring at positions \( [i, j) \) removed. So \( m = \def \ y < |x| \) and \( M \) accepts \( y \). In fact,

\[
\text{space } M(y) = \text{space } M(x) = f(n) > f(m) \geq \text{space } M(y),
\]
a contradiction. (To see the 1st equality, note that the space \( M \) uses is defined to be monotone w.r.t. time; or equivalently we could assume w.l.o.g. that \( M \) never writes a blank symbol.)

Next, \( \text{SPACE}(O(1)) \subseteq \text{SPACE}(0) \) holds since a constant amount of memory can be held in the state of a Turing machine.

Next we show that \( \text{SPACE}(0) \subseteq \text{the regular languages} \). Let \( M \) be a read-only TM with state set \( Q \). Here we will use a slightly different notion for the \( i \)th crossing sequence: it will include the sequence of states when the tape head of \( M \) is in either position \( i \) or \( i+1 \) and a note for each element in the sequence whether it represents the head being in position \( i \) or \( i+1 \). If on input \( x \) a crossing sequence has length \( > 2|Q| \), then \( M(x) \) does not halt. So \( M(x) \) accepts iff there are crossing sequences \( s_1, \ldots, s_{n-1} \) each of length \( \leq 2|Q| \) that are consistent with \( x \) and with each other and that halt in an accept state.

We construct an NFA \( N \) to decide whether such crossing sequences exist. \( N \) will scan \( x \) from left to right and guess a crossing sequence at each step, remembering the most recent 2 guesses to compare them for consistency. The first sequence must begin with the start state of \( M \). If some sequence has an accept state, then \( N \) will remember this and accept once it has seen a crossing sequence where the tape head does not move right, since this will signify that the entire path of the movement of the tape head of \( M \) has been determined.

To describe the algorithm that \( N \) uses to determine whether 2 adjacent crossing sequences are consistent is trivial but tedious, so we skip it.

Finally, that the regular languages are contained in \( \text{SPACE}(\lg \lg \lg n) \) is obvious.