538: Scribble of 25th January 2000

By Jarl Friis

1 speed-up theorem

Let r be a space-constructible function such that $r(n) \geq n$. There exist a language $A \subseteq 1^*$ that satisfies the following. If M_i is a Turing machine satisfying $L(M_i) = A$, then there exist a (more space efficient) Turing machine M_j satisfying $L(M_j) = A$, such that $r(s_j(1^n)) \leq s_i(1.)$. Here s_i and s_j are space bounds for M_i and M_j respectively.

The statement also holds if s_i and s_j are replaced by time bounds for M_i and M_j respectively. Essentially this theorem says that there exists no optimal algorithm for certain problems.

1.1 proof:

Let s be a space-constructible function such that $s(n) \geq r(n)$. Let h be a function defined as $h(0) = 2, \forall n \geq 0 : h(n+1) = s(h(n))$, so $h(n)$ is s composed n times applied to 2.

We want to show existence of machine M_j in the theorem. Therefore define A to be the language decided by SPACEY.

Algorithm 1: SPACEY

```
INPUT a string 1^n.
OUTPUT (accept/reject, cancelled)
if n = 1 then cancelled := ∅
else call SPACEY(1^{n-1}) to compute cancelled   {ignore accept/reject}
Find the smallest j, such that
      j ≤ n and j ∉ cancelled and s_j(1^n) < h(n - j)
if no such j exist then return (accept, cancelled)   {accept or reject
      doesn't matter}
else
   cancelled := cancelled ∪ {j}
   if M_j(1^n) accepts then return (reject, cancelled)
   if M_j(1^n) rejects then return (accept, cancelled)
```

The language A satisfies two properties:

1. $\forall k : A \in \mathrm{DSPACE}(h(n-k))$
2. $L(M_i) = A \Rightarrow s_i(1^n) \geq h(n-i)$ for all large n

Property 2 follows from contraposition, so we show that $\overset{\infty}{\exists} s_i(1^n) < h(n-i) \Rightarrow L(M_i) \neq A$, but this is exactly what is ensured in line 9-10, which are only executed if $s_i(1^n) < h(n-i)$

In line 5 we test if $s_j(1^n) < h(n-j)$, one way to do this is to do a simulation of M_j to see if the space used by M_j is less than $h(n-j)$. But with a restriction of not using more than $h(n-k)$ space this simulation cannot be done if $j \leq k$.

We solve this problem by using a table, for a given k we use a table containing the truth value of $s_j(1^n) < h(n-j)$ for all values of $j \leq k$, further more if $s_j(1^n) < h(n-j)$, in this table we also store the information concerning acceptance or rejection. For a given k we have a corresponding machine.

Now given such a table the test $s_j(1^n) < h(n-j)$ in line 5 is done by either looking up in the table, or performing a simulation of M_j on input 1^n, depending on whether $j \leq k$ or not.

Now we can with this use see that for any given k Spacey can decide A in space $h(n-k)$, hence proposition 1 holds.

We now see that there exist an algorithm is "less space consuming" than any other machine. We have the following inequalities:

$$r(s_j(1^n)) \leq r(h(n-i-1)) \leq s(h(n-i-1)) = h(n-i) \leq s_i(1^n)$$

Where s_i is the space bound of the machine M_i, s_j is the space bound on machine M_j which is designed from our algorithm for $k = i+1$. The first inequality comes from design of M_j, the second inequality comes from choice of s, the last one comes from proposition 2.

Fact: There exist a $B(n)$, such that for any (crazy) language A, and for any spaceconstructible function $s(n)$: $A \in \mathrm{DSPACE}(s(n)) \Leftrightarrow \Omega(B(n))$