CS538, Spring 1998
Script for Lecture on 2/2/98
by Takahiro Murata

Speed-up Theorem

This theorem shows that there is a problem for which *no* algorithm is remotely close to optimal.

> Let r be a total recursive function ($r(n) \geq n$). Then, there exists $A \subseteq 0^*$ such that
>
> If M_i is any Turing machine such that $L(M_i) = A$, then there exists M_j such that:
>
> $L(M_j) = A$ and
> $r(s_j(0^n)) \leq s_i(0^n)$ for all large n.

Proof: Let s be a space-constructible function such that $s(n) \geq max(r(n), n)$. We can build such $s(n)$ as follows:

```
on input x
    compute |x|(= n);
    compute r(n) while watching the space being used;
    mark off r(n) and n;
    compare these to the space used while computing r(n) and
      return the maximum of these
```

Let h be defined inductively as follows:

$h(0) = 2,$
$h(n+1) = s(h(n)).$

Note that $h(n) = s^{(n)}(2)$. (n-time composition of s applied to 2.)
Let A be the language recognized by the following machine:

```
Begin
on input 0^n
    if n = 1 then "cancelled" := φ;
    else call A(0^{n-1}) to compute "cancelled" (⊆ {1, ..., n});
    j := 1;
    while (j ≤ n and ¬(j ∉ "cancelled" and s_j(0^n) < h(n − j)) )
            // space h(n − j)(≤ h(n)) used
    j := j + 1
    end-while;
    if j > n then reject      // this is arbitrary (could be 'accept')
    else     // at this point, j ∉ "cancelled" and s_j(0^n) < h(n − j)
       "cancelled" := "cancelled" ∪ {j};
       accept ⟺ (M_j(0^n) ≠ accept)
End
```

Note that in the program description above, "call $A(0^{n-1})$" stands for a recursive call to the program with the input of length less than the original by one. We will see the following claims hold:

Claim 1 $\forall k, A \in DSPACE(h(n-k))$.

Claim 2 If $L(M_i) = A$, then $s_i(0^n) \geq h(n-i)$.

By **Claim 2** and how we have constructed s and h, for any M_i that recognizes A, for all large n:

$$r(h(n-i-1)) \leq s(h(n-i-1)) = h(n-i) \leq s_i(0^n)$$

However, by **Claim 1**, there is also a machine M_j to recognize A in space $h(n-i-1)$; i.e., $s_j(0^n) \leq h(n-i-1)$. Hence, the claim of the theorem. (r is assumed to be monotonic for all large inputs.)

To show **Claim 1**:
Prepare a table showing:

```
for all j ≤ k,
    the complete list of information concerning
    when TMs get cancelled (values of n, j, s_j(0^n), etc.)
      up to the last time any j ≤ k gets cancelled.
```

Note that we are *not* saying that it is possible to build such a table for a fixed k. Our claim is that one *does exist*, and (hypothetically) by using such a table, we argue as follows:

Let r be the input length on which the last $j \leq k$ gets cancelled. Revise the program above as follows:

```
Begin
on input 0^n
    if n ≤ r then look up the table to set "cancelled";
    else call A(0^{n-1}) to compute "cancelled" (⊆ {1, ..., n});
    j := 1;
    while (j ≤ n and ¬(j ∉ "cancelled" and s_j(0^n) < h(n − j)) )
            // note: for j ≤ k, we look up the table, so space ≤ h(n − k) used
    j := j + 1
    end-while;
    if j > n then reject
    else
       "cancelled" := "cancelled" ∪ {j};
       accept ⟺ (M_j(0^n) ≠ accept)
End
```

We can observe that this version of program uses space $h(n-k)$. □

To show **Claim 2**:
Suppose that it is not the case. Then, for a large $x = 0^n$, i will be added to "cancelled." But, the above program does not agree with M_i as to accepting/rejecting such x. Hence, a contradiction. □