SPACE AND TIME HIERARCHIES

This material is covered in section 9.1 in the textbook.

We have seen that comparing the power of deterministic and nondeterministic time bounds is very challenging. For corresponding questions dealing with space bounds we have the nice result from Savitch's theorem. Next we consider the natural question: suppose we increase ("sufficiently much") the time or space bound, does this guarantee that the power of the (deterministic) Turing machines increases?

Our intuition suggests that giving Turing machines more time or space should increase the computational capability of the machines. Under some reasonable assumptions, this turns out to be the case: we can establish *strict* inclusions for the classes, assuming that we restrict consideration to *constructible* functions.

Definition. (Def. 9.1) A function $f : \mathbb{N} \longrightarrow \mathbb{N}$ is **space constructible** if there is a Turing machine that on input 1^n writes the binary representation of $f(n)$ on the output tape and the computation uses space $O(f(n))$.

In the above definition we always assume that $f(n) \geq \log n$ and hence it does not matter whether or not the space used on the output tape is counted.

- All the commonly used space bounds, such as polynomials, $\log n$, 2^n, are space constructible. This can be easily verified (examples in class).
  Note that in the definition of space constructibility of $f(n)$, the argument for the function is <u>represented in unary</u> – this corresponds to our usual convention that the argument of a space bound function is the length of the input.

Space hierarchy theorem (Th. 9.3) For any space constructible function $f : \mathbb{N} \longrightarrow \mathbb{N}$ there exists a language A decidable in space $O(f(n))$ but not decidable in space $o(f(n))$.

Proof: Uses diagonalization. Time permitting, we may go through the proof in class.

Corollary. For any functions $f_1(n), f_2(n) \geq \log n$ where

1. $f_1(n) = o(f_2(n))$, and
2. f_2 is space constructible

we have

$$\text{SPACE}(f_1(n)) \subset \text{SPACE}(f_2(n)).$$

Note that above "$\subset$" denotes strict inclusion.

Using Savitch's theorem we obtain also:

Corollary. NL $\subset$ SPACE(n)

Similarly if we define:

$$\text{EXPSPACE} = \bigcup_{k \geq 1} \text{SPACE}(2^{n^k})$$

the space hierarchy theorem gives the strict inclusion

$$\text{PSPACE} \subset \text{EXPSPACE}.$$

For establishing a hierarchy result for time complexity classes we again need the following technical definition.

Definition. (Def. 9.8) A function $t : \mathbb{N} \longrightarrow \mathbb{N}$, $t(n) \geq n$, is **time constructible** if there is a Turing machine that on input 1^n writes the binary representation of $t(n)$ on the tape and the computation uses $O(t(n))$ steps.

Time hierarchy theorem. (Th. 9.10) For any time constructible function $t(n) \geq n \cdot \log n$ there exists a language B decidable in time $O(t(n))$ but not in time $o(t(n)/\log t(n))$.

Corollary. $\text{TIME}(t_1(n)) \subset \text{TIME}(t_2(n))$ if $t_1(n) = o(t_2(n)/\log(t_2(n)))$ and $t_2(n) \geq n \cdot \log n$ is time constructible.

Corollary. $\text{P} \subset \text{EXPTIME}$

Why do we need the constructibility assumptions? If we would allow arbitrary functions as time/space bounds (or even arbitrary computable functions) it is possible to get very strange results of the following type:

Gap theorem (Borodin, 1972) There is a total strictly monotonic computable function $f : \mathbb{N} \longrightarrow \mathbb{N}$ such that

$$\text{TIME}(f(n)) = \text{TIME}(2^{f(n)}).$$

The "gap theorem" is proved using a form of diagonalization (but the proof is beyond the scope of this course). The reasons for the constructibility assumptions are discussed also in the early part of section 9.1 in the textbook.

By adding the constructibility assumption we can eliminate the strange situations implied by the gap theorem, and we are able to establish the time and space hierarchy results discussed above. All reasonable ("naturally occurring") functions that are at least linear are normally time constructible, and all reasonable functions that are at least logarithmic are normally space constructible.

Examples. What are the relationships between the following pairs of complexity classes? (equal, strict inclusion, inclusion that is not known to be strict)

1. $\text{TIME}(2^n)$ and $\text{TIME}(2^{n+5})$
2. $\text{TIME}(2^n)$ and $\text{TIME}(3^n)$
3. $\text{NSPACE}(2^n)$ and $\text{SPACE}(5^n)$

4. NSPACE(2^n) and SPACE(4^n)

5. SPACE(n) and SPACE($n \cdot \log n$)

6. TIME(n) and TIME($n \cdot \log n$)

7. TIME(2^n) and TIME($n^2 \cdot 2^n$)