Complexity Theory
Lecture 12

Anuj Dawar

University of Cambridge Computer Laboratory
Easter Term 2011

http://www.cl.cam.ac.uk/teaching/1011/Complexity/

Complexity Classes

We have established the following inclusions among complexity classes:

$$\mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE} \subseteq \mathsf{EXP}$$

Showing that a problem is NP-complete or PSPACE-complete, we often say that we have proved it intractable.

While this is not strictly correct, a proof of completeness for these classes does tell us that the problem is structurally difficult.

Similarly, we say that PSPACE-complete problems are harder than NP-complete ones, even if the running time is not higher.

Logarithmic Space Reductions

We write

$$A \leq_L B$$

if there is a reduction f of A to B that is computable by a deterministic Turing machine using $O(\log n)$ workspace (with a *read-only* input tape and *write-only* output tape).

Note: We can compose $\leq_L$ reductions. So,

$$\text{if } A \leq_L B \text{ and } B \leq_L C \text{ then } A \leq_L C$$

NP-complete Problems

Analysing carefully the reductions we constructed in our proofs of NP-completeness, we can see that SAT and the various other NP-complete problems are actually complete under $\leq_L$ reductions.

Thus, if $\mathsf{SAT} \leq_L A$ for some problem in L then not only $\mathsf{P} = \mathsf{NP}$ but also $\mathsf{L} = \mathsf{NP}$.

P-complete Problems

It makes little sense to talk of complete problems for the class P with respect to polynomial time reducibility $\leq_P$.

There are problems that are complete for P with respect to *logarithmic space* reductions $\leq_L$.

One example is CVP—the circuit value problem.

- If CVP $\in$ L then L $=$ P.
- If CVP $\in$ NL then NL $=$ P.

Provable Intractability

Our aim now is to show that there are languages (*or, equivalently, decision problems*) that we can prove are not in P.

This is done by showing that, for every *reasonable* function f, there is a language that is not in TIME$(f(n))$.

The proof is based on the diagonal method, as in the proof of the undecidability of the halting problem.

Constructible Functions

A complexity class such as TIME$(f(n))$ can be very unnatural, if $f(n)$ is.

We restrict our bounding functions $f(n)$ to be proper functions:

Definition
A function $f : \mathbb{N} \rightarrow \mathbb{N}$ is *constructible* if:

- f is non-decreasing, i.e. $f(n+1) \geq f(n)$ for all n; and
- there is a deterministic machine M which, on any input of length n, replaces the input with the string $0^{f(n)}$, and M runs in time $O(n + f(n))$ and uses $O(f(n))$ *work space.*

Examples

All of the following functions are constructible:

- $\lceil \log n \rceil$;
- n^2;
- n;
- 2^n.

If f and g are constructible functions, then so are $f + g$, $f \cdot g$, 2^f and $f(g)$ (this last, provided that $f(n) > n$).

Using Constructible Functions

NTIME($f(n)$) can be defined as the class of those languages L accepted by a *nondeterministic* Turing machine M, such that for every $x \in L$, there is an accepting computation of M on x of length at most $O(f(n))$.

If f is a constructible function then any language in NTIME($f(n)$) is accepted by a machine for which all computations are of length at most $O(f(n))$.

Also, given a Turing machine M and a constructible function f, we can define a machine that simulates M for $f(n)$ steps.

Inclusions

The inclusions we proved between complexity classes:

- NTIME($f(n)$) $\subseteq$ SPACE($f(n)$);
- NSPACE($f(n)$) $\subseteq$ TIME($k^{\log n + f(n)}$);
- NSPACE($f(n)$) $\subseteq$ SPACE($f(n)^2$)

really only work for *constructible* functions f.

The inclusions are established by showing that a deterministic machine can simulate a nondeterministic machine M for $f(n)$ steps.

For this, we have to be able to compute f within the required bounds.

Time Hierarchy Theorem

For any constructible function f, with $f(n) \geq n$, define the f-bounded *halting language* to be:

$$H_f = \{[M], x \mid M \text{ accepts } x \text{ in } f(|x|) \text{ steps}\}$$

where $[M]$ is a description of M in some fixed encoding scheme.

Then, we can show

$H_f \in$ TIME($f(n)^3$) and $H_f \notin$ TIME($f(\lfloor n/2 \rfloor)$)

Time Hierarchy Theorem
For any constructible function $f(n) \geq n$, TIME($f(n)$) is properly contained in TIME($f(2n+1)^3$).

Strong Hierarchy Theorems

For any constructible function $f(n) \geq n$, TIME($f(n)$) is properly contained in TIME($f(n)(\log f(n))$).

Space Hierarchy Theorem
For any pair of constructible functions f and g, with $f = O(g)$ and $g \neq O(f)$, there is a language in SPACE($g(n)$) that is not in SPACE($f(n)$).

Similar results can be established for nondeterministic time and space classes.

Consequences

- For each k, $\mathsf{TIME}(n^k) \neq \mathsf{P}$.
- $\mathsf{P} \neq \mathsf{EXP}$.
- $\mathsf{L} \neq \mathsf{PSPACE}$.
- Any language that is EXP-complete is not in P.
- There are no problems in P that are complete under linear time reductions.

The End

Please provide *feedback*:

`https://camtools.cam.ac.uk/direct/eval-evaluation/4681`