# Complexity Theory
# Lecture 11

Anuj Dawar

University of Cambridge Computer Laboratory
Easter Term 2010

http://www.cl.cam.ac.uk/teaching/0910/Complexity/





## Inclusions

We have the following inclusions:

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

where $\mathsf{EXP} = \bigcup_{k=1}^{\infty} \mathsf{TIME}(2^{n^k})$

Moreover,

$$\mathsf{L} \subseteq \mathsf{NL} \cap \text{co-}\mathsf{NL}$$

$$\mathsf{P} \subseteq \mathsf{NP} \cap \text{co-}\mathsf{NP}$$

$$\mathsf{PSPACE} \subseteq \mathsf{NPSPACE} \cap \text{co-}\mathsf{NPSPACE}$$





## Establishing Inclusions

To establish the known inclusions between the main complexity classes, we prove the following.

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

The first two are straightforward from definitions.

The third is an easy simulation.

The last requires some more work.





## Reachability

Recall the Reachability problem: given a *directed* graph $G = (V, E)$ and two nodes $a, b \in V$, determine whether there is a path from $a$ to $b$ in $G$.

A simple search algorithm solves it:

1. mark node $a$, leaving other nodes unmarked, and initialise set $S$ to $\{a\}$;
2. while $S$ is not empty, choose node $i$ in $S$: remove $i$ from $S$ and for all $j$ such that there is an edge $(i, j)$ and $j$ is unmarked, mark $j$ and add $j$ to $S$;
3. if $b$ is marked, accept else reject.

## NL Reachability

We can construct an algorithm to show that the Reachability problem is in NL:

1. write the index of node $a$ in the work space;
2. if $i$ is the index currently written on the work space:
   (a) if $i = b$ then accept, else guess an index $j$ ($\log n$ bits) and write it on the work space.
   (b) if $(i, j)$ is not an edge, reject, else replace $i$ by $j$ and return to (2).





We can use the $O(n^2)$ algorithm for Reachability to show that:

$\mathsf{NSPACE}(f(n)) \subseteq \mathsf{TIME}(k^{\log n + f(n)})$

for some constant $k$.

Let $M$ be a nondeterministic machine working in space bounds $f(n)$.

For any input $x$ of length $n$, there is a constant $c$ (depending on the number of states and alphabet of $M$) such that the total number of possible configurations of $M$ within space bounds $f(n)$ is bounded by $n \cdot c^{f(n)}$.

> Here, $c^{f(n)}$ represents the number of different possible contents of the work space, and $n$ different head positions on the input.





## Configuration Graph

Define the *configuration graph* of $M, x$ to be the graph whose nodes are the possible configurations, and there is an edge from $i$ to $j$ if, and only if, $i \rightarrow_M j$.

Then, $M$ accepts $x$ if, and only if, some accepting configuration is reachable from the starting configuration $(s, \triangleright, x, \triangleright, \varepsilon)$ in the configuration graph of $M, x$.





Using the $O(n^2)$ algorithm for Reachability, we get that $M$ can be simulated by a deterministic machine operating in time

$$c'(nc^{f(n)})^2 \sim c'c^{2(\log n + f(n))} \sim k^{(\log n + f(n))}$$

In particular, this establishes that $\mathsf{NL} \subseteq \mathsf{P}$ and $\mathsf{NPSPACE} \subseteq \mathsf{EXP}$.

## Savitch's Theorem

Further simulation results for nondeterministic space are obtained by other algorithms for Reachability.

We can show that Reachability can be solved by a *deterministic* algorithm in $O((\log n)^2)$ space.

Consider the following recursive algorithm for determining whether there is a path from $a$ to $b$ of length at most $n$ (for $n$ a power of 2):





$O((\log n)^2)$ space Reachability algorithm:

Path$(a, b, i)$

if $i = 1$ and $a \neq b$ and $(a, b)$ is not an edge reject
else if $(a, b)$ is an edge or $a = b$ accept
else, for each node $x$, check:

1. is there a path $a - x$ of length $i/2$; and
2. is there a path $x - b$ of length $i/2$?

if such an $x$ is found, then accept, else reject.

The maximum depth of recursion is $\log n$, and the number of bits of information kept at each stage is $3 \log n$.





## Savitch's Theorem - 2

The space efficient algorithm for reachability used on the configuration graph of a nondeterministic machine shows:

$$\mathsf{NSPACE}(f(n)) \subseteq \mathsf{SPACE}(f(n)^2)$$

for $f(n) \geq \log n$.

This yields

$$\mathsf{PSPACE} = \mathsf{NPSPACE} = \mathsf{co\text{-}NPSPACE}.$$





## Complementation

A still more clever algorithm for Reachability has been used to show that nondeterministic space classes are closed under complementation:

If $f(n) \geq \log n$, then

$$\mathsf{NSPACE}(f(n)) = \mathsf{co\text{-}NSPACE}(f(n))$$

In particular

$$\mathsf{NL} = \mathsf{co\text{-}NL}.$$

# 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 SAT $\leq_L A$ for some problem in L then not only P = NP but also L = 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.