Complexity Theory	1	Complexity Theory	2	
Complexity Theory Lecture 11		Incl We have the following inclusions	usions 	
		$L \subseteq NL \subseteq P \subseteq NP \subseteq PS$ where $EXP = \bigcup_{k=1}^{\infty} TIME(2^{n^k})$	$SPACE \subseteq NPSPACE \subseteq EXP$	
Anuj Dawar		where $E_{k} = O_{k=1} + O_{k=1}$		
University of Cambridge Computer Laboratory		Moreover,		
Easter Term 2013		$L \subseteq NL \cap co-NL$		
		$P \subseteq NP \cap co-NP$ $PSPACE \subseteq NPSPACE \cap co-NPSPACE$		
http://www.cl.cam.ac.uk/teaching/1213/Complexit	May 20, 2013	Anuj Dawar	May 20, 2013	
Complexity Theory	3	Complexity Theory	4	
Establishing Inclusions			hability	
To establish the known inclusions between the main complex classes, we prove the following.			Recall the Reachability problem: given a <i>directed</i> graph $G = (V, E)$ and two nodes $a, b \in V$, determine whether there is a path from a to b in G .	
• $SPACE(f(n)) \subseteq NSPACE(f(n));$				
• $TIME(f(n)) \subseteq NTIME(f(n));$		A simple search algorithm solves	s it:	
• $NTIME(f(n)) \subseteq SPACE(f(n));$		1. mark node a , leaving other nodes unmarked, and initialise set		
• NSPACE $(f(n)) \subseteq TIME(k^{\log n + f(n)});$		S to $\{a\};$		
The first two are straightforward from definitions. The third is an easy simulation.		 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; 		
The last requires some more work.		3. if b is marked, accept else re	ject.	

5

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).

Anuj Dawar

Complexity Theory

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, $i \to_M j$.

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.

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.

Complexity Theory

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.

Anuj Dawar

May 20, 2013

8

Complexity Theory

Using the $O(n^2)$ algorithm for Reachability, we get that L(M)—the language accepted by M—can be decided 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 $NL \subseteq P$ and $NPSPACE \subseteq EXP$.

Anuj Dawar

May 20, 2013

7

9

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 i (for i a power of 2):

Complexity Theory

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

Path(a, b, i)

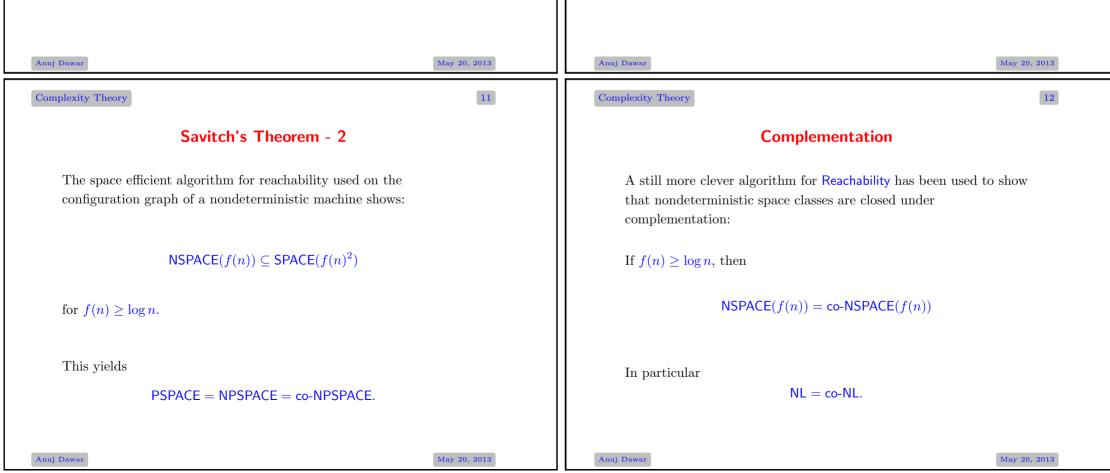
if i = 1 and $a \neq b$ and (a, b) is not an edge reject else if (a, b) is an edge or a = b accept else, for each node x, check:

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 $3 \log n$.



We write

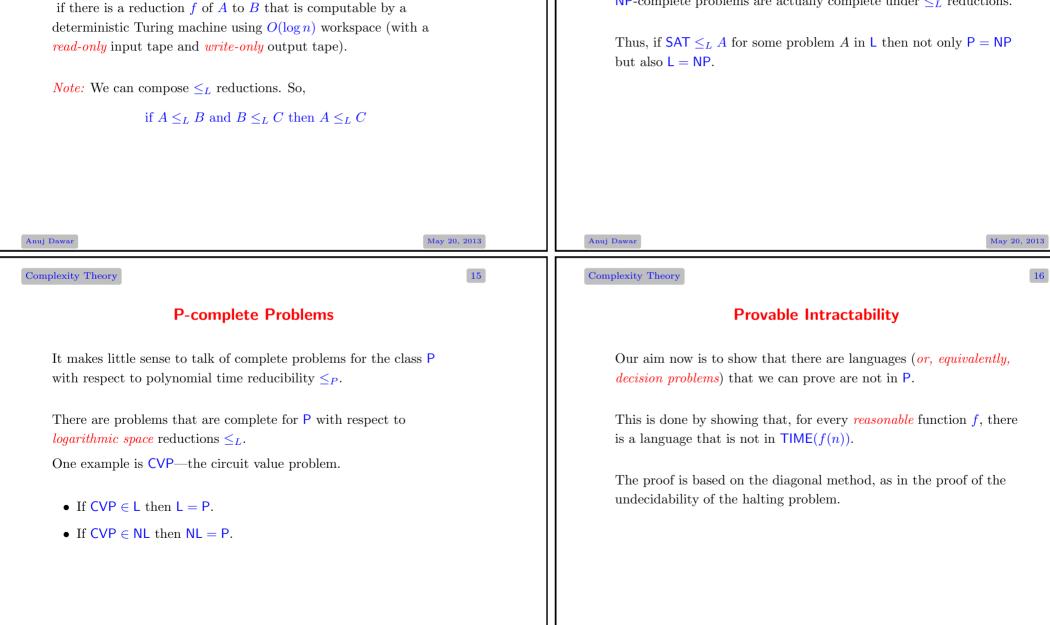
Logarithmic Space Reductions

 $A \leq_L B$

13

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.



f(n) is.

Definition

Constructible Functions

A complexity class such as $\mathsf{TIME}(f(n))$ can be very unnatural, if

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

• f is non-decreasing, i.e. $f(n+1) \ge f(n)$ for all n; and

in time O(n + f(n)) and uses O(f(n)) work space.

• 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

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

17

18

All of the following functions are constructible:

• $\lceil \log n \rceil;$

• $n^2;$

• *n*;

• 2ⁿ.

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).

Anuj Dawar	May 20, 2013	Anuj Dawar May 20, 2013	
Complexity Theory	19	Complexity Theory 20]
Using Constructible Functions		Inclusions	
NTIME $(f(n))$ can be defined as the class of those language accepted by a <i>nondeterministic</i> Turing machine M , such a every $x \in L$, there is an accepting computation of M on x length at most $O(f(n))$. If f is a constructible function then any language in NTIM is accepted by a machine for which all computations are of at most $O(f(n))$. Also, given a Turing machine M and a constructible func- can define a machine that simulates M for $f(n)$ steps.	that for c of ME(f(n)) of length	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 <i>constructible</i> 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.	