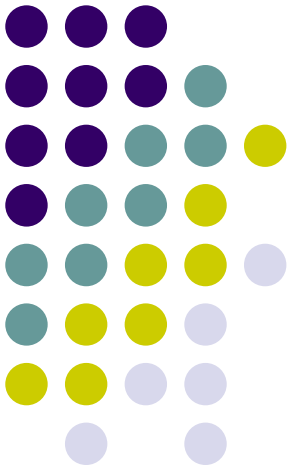
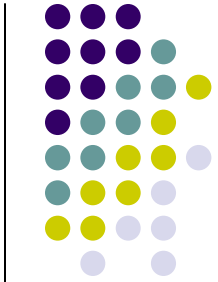
CS256
Applied Theory of Computation

Complexity Classes III

John E Savage



Overview



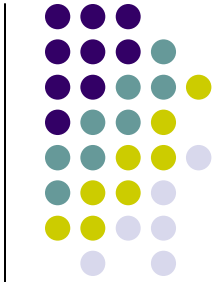
- Last lecture on time-bounded complexity classes
- Today we examine space-bounded complexity classes
- We prove *Savitch's Theorem*, i.e. NSPACE(r(n)), the set of languages recognized in non-deterministic space r(n), is contained in SPACE(r^2(n)).
- A consequence of Savitch's Theorem is that PSPACE = NPSPACE
- Nondeterminism doesn't increase the power of TMs when space used is polynomial length of the input!

Reductions Between Problems

Definition If L_1 and L_2 are langs., a **transformation** is a DTM computable function h: B*→B* such that **x** in L_1 if and only if h(**x**) is in L_2. A **resource-bounded transformation** is a transformation computed under a resource bound.

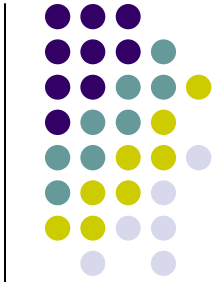
Definition For decision probs. P_1 and P_2, $P_1 <_R P_2$ if P_1 can be transformed to P_2 by a transformation in the class R.

Reductions



- **Definition** Let C be a complexity class, R a class of resource-bounded transformations, and P_1 and P_2 decision problems. A set of transformations R is compatible with C if $P_1 <_R P_2$ and P_2 in C implies that P_1 is in C.
- Poly-time transformations ($<_p$) are compatible with **P**
- **Definition** A class R of transformations is *transitive* if the composition of any two transformations in R is another in R and for all decision problems P_1, P_2, and P_3, $P_1 <_R P_2$, and $P_2 <_R P_3$ imply that $P_1 <_R P3$.

Logspace Transformations

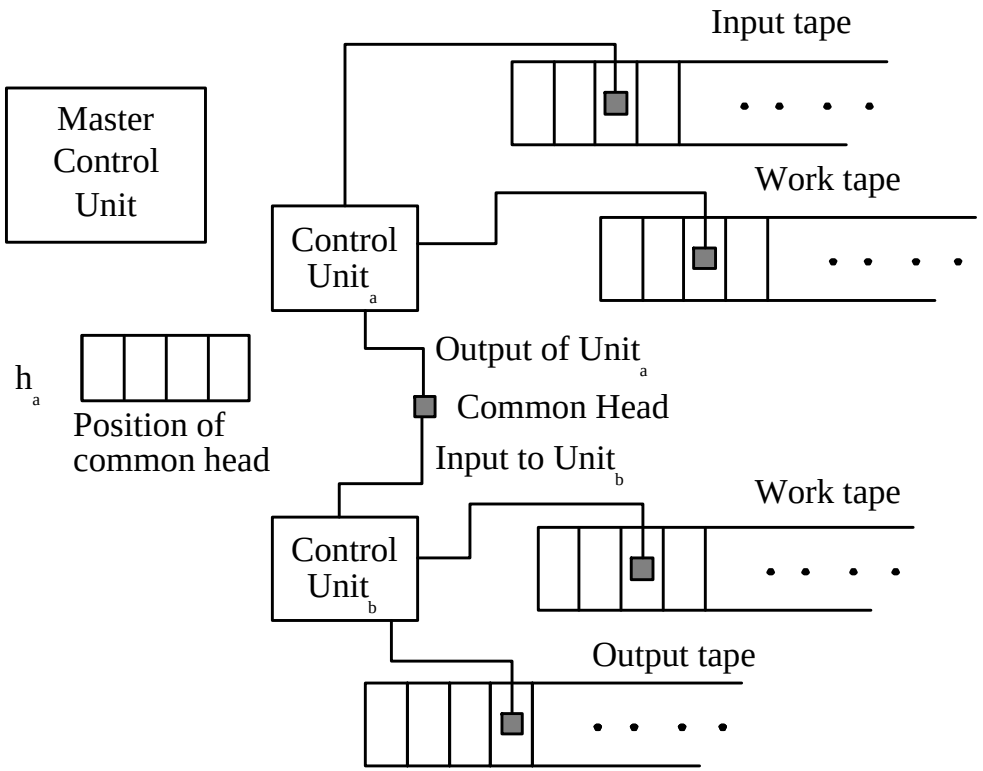


Theorem Log-space transformations (<log-space) are transitive.

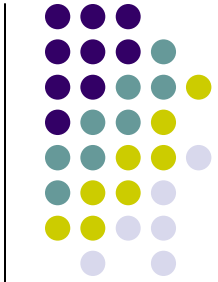
Proof Let M_a and M_b be DTMs computing transf. from P_1 to P_2 and transf. from P_2 to P_3. Let M simulate their composition. The output tape of M_a is the input tape of M_b. Eliminate it to avoid exceeding O(*log* n) storage limit of combined machine.

How can we replace M_a's output tape?

Logspace Transformations (cont.)



Logspace Transformations (cont.)



Eliminate M_a's output tape by replacing it with a counter and storage for a single input/output cell. The value N of the counter is the position of M_a's output head, which is M_b's input head, on their common tape. If M_b needs a new input, M simulates M_a until it produces a new output. If M_b needs M_a's previous output (the head moves back one cell), M restarts the simulation of M_a and simulates N-1 steps.

Logspace Transformations (cont.)

Earlier we counted the number of configurations of a Turing machine M with input and output tapes and s work tapes. We repeat this analysis. Let r(n) be the maximum number of tape cells used and let c be the maximal size of a tape alphabet. Then, M can be in one of at most $\chi \le c^{s\, r(n)} (n\, r(n)^s) = O(k^{r(n)})$ configurations for some k 1. Since M always halts, by the pigeonhole principle, it passes through at most c configurations in at most c steps. Because r(n) = O(*log* n), $\chi = O(n^d)$ for some integer d. Thus, M executes a polynomial number of steps.

Logspace Transformations (cont.)

- This simulation uses O(*log* n) space to simulate the composition of the two O(*log* n) transformations. Q.E.D.

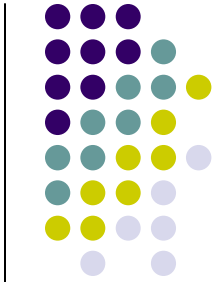
Hard and Complete Problems

Definition Let R be a class of reductions, C a complexity class, and R compatible with C. A **problem *Q* is hard for C under R-reductions** if for every problem in *P* in C, $P <_R Q$. *Q* is **complete** for C under R-reductions if *Q* is also in C.

Definition Problems in **P** hard for **P** under log-space reductions are **P**-complete. Problems in **NP** hard for **NP** under polynomial-time reductions are **NP**- complete.

Theorem If a **P**-complete problem can be solved in log-space, so can all problems in **P**. If an **NP**-complete prob. is in **P**, then **P=NP**

A Complete Problem for P



CIRCUIT VALUE

Instance: A circuit description with fixed values for inputs and a designated circuit output.

Answer: "Yes" if the output has value 1.

This problem is shown in Chpt. 3 to be **P**-complete.

LINEAR INEQUALITIES

Instance: Integer valued m×n matrix A and column m-vector **b**

Answer: "Yes" if there is a rational column vector $\mathbf{x} > 0$ such that $A\mathbf{x} \leq \mathbf{b}$.

A Complete Problem for P (cont.)

Theorem LINEAR INEQUALITIES is **P**-complete.

Proof Reduced from CIRCUIT VALUE. Replace each gate or assignment by inequalities:

u=1 by $u \le 1$ and $1 \le u$.

u=0 by $u \le 0$ and $0 \le u$.

$w = u \wedge v$ by $0 \le w \le 1$, $w \le u$, $w \le v$, $u+v-1 \le w$.

$w = u \vee v$ by $0 \le w \le 1$, $u \le w$, $v \le w$, $w \le u+v-1$.

$w = \text{NOT } u$ by $0 \le w \le 1$, $w \le 1-u$, $1-u \le w$.

A Complete Problem for P (cont.)

Given an instance of circuit value of length n with O(*log* n) space we use these inequalities to transform it to an instance of linear inequalities. By the nature of the construction, an instance of the first is a “Yes” instance if and only if an instance of the latter is a “Yes” instance. Q.E.D.

Complete Problems for NP

In book it is shown that circuit sat, satisfiability, 3-sat, naesat, 3-coloring, and set cover are all **NP**-complete.

INDEPENDENT SET

Instance: A graph G = (V,E) and an integer k.

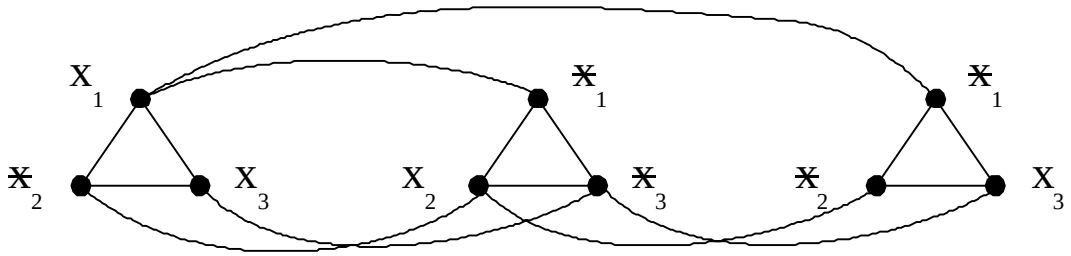
Answer: "Yes" if there are k vertices of G such that there is no edge in E between them.

NP-Complete Problem

Theorem INDEPENDENT SET is **NP**-complete

Proof INDEPENDENT SET is in **NP**. Reduce 3-sat to it. Each 3-sat clause has 3 literals (easy to fix if not). 3-SAT example:

$$(x_1+x_2+x_3)\,(x_1+x_2+x_3)\,(x_1+x_2+x_3)$$



NP-Complete Problem

Proof (cont.) Construct an instance of independent set with k = # clauses. (In the example, k = 3.) Build triangle for each clause; attach name of literal to its vertex.

Add edge between complementary literals in different clauses. (Ensures that both literals can't be in an independent set.) Easy log-space reduction. How do we show an instance of one decision problem is a "Yes" instance if and only if it is a "Yes" instance of the other?

Boundary Between P and NP

2-SAT same as 3-SAT except for at most two literals per clause.

Theorem 2-SAT is in NL.

Proof See page 363 and 364, revised.