Topics in Logic and Complexity
Handout 2

Anuj Dawar

MPhil Advanced Computer Science, Lent 2010

Polynomial Time Computation

$$\mathsf{P} = \bigcup_{k=1}^{\infty} \mathsf{TIME}(n^k)$$

The class of languages decidable in polynomial time.

The complexity class P plays an important role in complexity theory.

- It is robust, as explained.
- It serves as our formal definition of what is *feasibly computable*

Nondeterministic Polynomial Time

$$\mathsf{NP} = \bigcup_{k=1}^{\infty} \mathsf{NTIME}(n^k)$$

That is, NP is the class of languages accepted by a *nondeterministic* machine running in polynomial time.

Since a deterministic machine is just a nondeterministic machine in which the transition relation is *functional*, $\mathsf{P} \subseteq \mathsf{NP}$.

Succinct Certificates

The complexity class NP can be characterised as the collection of languages of the form:

$$L = \{x \mid \exists y R(x, y)\}$$

Where R is a relation on strings satisfying two key conditions

1. R is decidable in polynomial time.
2. R is *polynomially balanced*. That is, there is a polynomial p such that if $R(x, y)$ and the length of x is n, then the length of y is no more than $p(n)$.

Equivalence of Definitions

If $L = \{x \mid \exists y\ R(x, y)\}$ we can define a nondeterministic machine M that accepts L.

The machine first uses nondeterministic branching to *guess* a value for y, and then checks whether $R(x, y)$ holds.

In the other direction, suppose we are given a nondeterministic machine M which runs in time $p(n)$.

Suppose that for each $(q, \sigma) \in K \times \Sigma$ (i.e. each state, symbol pair) there are at most k elements in $\delta(q, \sigma)$.

Equivalence of Definitions

For y a string over the alphabet $\{1, \ldots, k\}$, we define the relation $R(x, y)$ by:

- $|y| \leq p(|x|)$; and
- the computation of M on input x which, at step i takes the "$y[i]$th transition" is an accepting computation.

Then, $L(M) = \{x \mid \exists y\ R(x, y)\}$

Space Complexity Classes

$\mathsf{L} = \mathsf{SPACE}(\log n)$
The class of languages decidable in logarithmic space.

$\mathsf{NL} = \mathsf{NSPACE}(\log n)$
The class of languages decidable by a nondeterministic machine in logarithmic space.

$\mathsf{PSPACE} = \bigcup_{k=1}^{\infty} \mathsf{SPACE}(n^k)$
The class of languages decidable in polynomial space.

$\mathsf{NPSPACE} = \bigcup_{k=1}^{\infty} \mathsf{NSPACE}(n^k)$

Inclusions between Classes

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

Of these, the following are direct from the definitions:

$$\mathsf{L} \subseteq \mathsf{NL}$$

$$\mathsf{P} \subseteq \mathsf{NP}$$

$$\mathsf{PSPACE} \subseteq \mathsf{NPSPACE}$$

NP $\subseteq$ PSPACE

To simulate a nondeterministic machine M running in time $t(n)$ by a deterministic one, it suffices to carry out a *depth-first* search of the computation tree.

We keep a counter to cut off branches that exceed $t(n)$ steps.

The space required is:

- a *counter* to count up to $t(n)$; and
- a *stack* of configurations, each of size at most $O(t(n))$.

The depth of the stack is at most $t(n)$.

Thus, if t is a polynomial, the total space required is polynomial.

NL $\subseteq$ P

Given a nondeterministic machine M that works with *work space* bounded by $s(n)$ and an input x of length n, there is some constant c such that

> the total number of possible configurations of M within space bounds $s(n)$ is bounded by $n \cdot c^{s(n)}$.

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, $i \rightarrow_M j$.

Reachability in the Configuration Graph

M accepts x if, and only if, some accepting configuration is reachable from the starting configuration 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

$$c'(nc^{s(n)})^2 \sim c'c^{2(\log n + s(n))} \sim d^{(\log n + s(n))}$$

for some constant d.

When $s(n) = O(\log n)$, this is polynomial and so NL $\subseteq$ P.

When $s(n)$ is polynomial this is exponential in n and so NPSPACE $\subseteq$ EXP.

Nondeterministic Space Classes

If *Reachability* were solvable by a *deterministic* machine with logarithmic space, then

$$\text{L} = \text{NL}.$$

In fact, *Reachability* is solvable by a deterministic machine with space $O((\log n)^2)$.

This implies

$$\text{NSPACE}(s(n)) \subseteq \text{SPACE}((s(n)^2)).$$

In particular PSPACE = NPSPACE.

Reachability in $O((\log n)^2)$

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

Path(a, b, i)

if $i = 1$ 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 $3 \log n$.

Inclusions between Classes

This leaves us with the following:

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

Hierarchy Theorems proved by *diagonalization* can show that:

$$\mathsf{L} \neq \mathsf{PSPACE} \quad \mathsf{NL} \neq \mathsf{NPSPACE} \quad \mathsf{P} \neq \mathsf{EXP}$$

For other inclusions above, it remains an open question whether they are strict.

Complement Classes

If we interchange accepting and rejecting states in a deterministic machine that accepts the language L, we get one that accepts $\overline{L}$.

If a language $L \in \mathsf{P}$, then also $\overline{L} \in \mathsf{P}$.

Complexity classes defined in terms of nondeterministic machine models are not necessarily closed under complementation of languages.

Define,

co-NP – the languages whose complements are in NP.

co-NL – the languages whose complements are in NL.

Relationships

$\mathsf{P} \subseteq \mathsf{NP} \cap \text{co-}\mathsf{NP}$ and any of the situations is consistent with our present state of knowledge:

- $\mathsf{P} = \mathsf{NP} = \text{co-}\mathsf{NP}$
- $\mathsf{P} = \mathsf{NP} \cap \text{co-}\mathsf{NP} \neq \mathsf{NP} \neq \text{co-}\mathsf{NP}$
- $\mathsf{P} \neq \mathsf{NP} \cap \text{co-}\mathsf{NP} = \mathsf{NP} = \text{co-}\mathsf{NP}$
- $\mathsf{P} \neq \mathsf{NP} \cap \text{co-}\mathsf{NP} \neq \mathsf{NP} \neq \text{co-}\mathsf{NP}$

It follows from the fact that $\mathsf{PSPACE} = \mathsf{NPSPACE}$ that NPSPACE is closed under complementation.

Also, **Immerman and Szelepcsényi** showed that $\mathsf{NL} = \text{co-}\mathsf{NL}$.

Reductions

Given two languages $L_1 \subseteq \Sigma_1^\star$, and $L_2 \subseteq \Sigma_2^\star$,

A *reduction* of L_1 to L_2 is a *computable* function

$$f : \Sigma_1^\star \to \Sigma_2^\star$$

such that for every string $x \in \Sigma_1^\star$,

$$f(x) \in L_2 \text{ if, and only if, } x \in L_1$$

Resource Bounded Reductions

If f is computable by a polynomial time algorithm, we say that L_1 is *polynomial time reducible* to L_2.

$$L_1 \leq_P L_2$$

If f is also computable in $\mathsf{SPACE}(\log n)$, we write

$$L_1 \leq_L L_2$$

Reductions 2

If $L_1 \leq L_2$ we understand that L_1 is no more difficult to solve than L_2.

That is to say, for any of the complexity classes $\mathcal{C}$ we consider,

If $L_1 \leq L_2$ and $L_2 \in \mathcal{C}$, then $L_1 \in \mathcal{C}$

We can get an algorithm to decide L_1 by first computing f, and then using the $\mathcal{C}$-algorithm for L_2.

Provided that $\mathcal{C}$ is *closed* under such reductions.

Completeness

The usefulness of reductions is that they allow us to establish the *relative* complexity of problems, even when we cannot prove absolute lower bounds.

Cook (1972) first showed that there are problems in NP that are maximally difficult.

For any complexity class $\mathcal{C}$, a language L is said to be *$\mathcal{C}$-hard* if for every language $A \in \mathcal{C}$, $A \leq L$.

A language L is *$\mathcal{C}$-complete* if it is in $\mathcal{C}$ and it is $\mathcal{C}$-hard.

Complete Problems

Examples of complete problems for various complexity classes.

NL
Reachability

P
Game, Circuit Value Problem

NP Satisfiability of Boolean Formulas, Graph 3-Colourability, Hamiltonian Cycle

co-NP
Validity of Boolean Formulas, Non 3-colourability

PSPACE
Geography, The game of HEX

Reading List for this Handout

1. Papadimitriou. Chapters 7, 8 and 16.
2. Immerman Chapter 2.