E0 224 Computational Complexity Theory **Indian Institute of Science, Bangalore**
Fall 2014 **Department of Computer Science and Automation**

Lecture 12: Sep 15, 2014

Lecturer: Chandan Saha <chandan@csa.iisc.ernet.in> *Scribe: Sachin Kumar Srivastava*

12.1 Certificate Definition of the class NL

We define NL using certificates as we did it for NP instead of Non deterministic Turing Machines. But the certificate provided can be too long that the space required is much larger than log-space. We resolve this issue by considering the certificate to be read-once i.e. we can see any bit of the certificate only once rather than again and again. Now, we define NL using certificate.

A language $L \subseteq \{0,1\}^*$ is in class NL iff there is a polynomial function q(.) and a log-space Turing Machine M such that

$$x \in L \iff \exists u \in \{0,1\}^{q(|x|)} \text{ and } M(x,u) = 1,$$

where u is given in special "read once tape" to M, where by M (x, u) we denote the output of M where x is placed on its input tape and u is placed on its special read-once tape, and M uses at most O(log $|x|$) space on its read/write tapes for every input x.

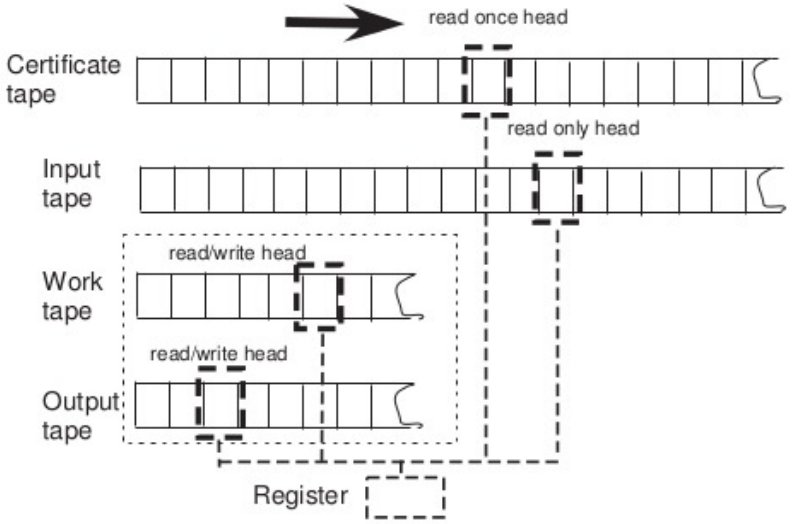

Figure 12.1: Certificate view of NL. The certificate for input x is placed on a special read-once tape on which the machines head can never move to the left (figure is taken from [1]).

Note : If we allow it to move left and right, it becomes NP.

Example of an NL-Complete problem is

$PATH$ = $\{< G, s, t > \mid G$ is a directed graph and there is a path from s to t in $G\}$

co-NL = $\{L | \overline{L} \in NL\}$

Remark : $\overline{PATH} = \{< G, s, t > \mid G$ is a directed graph and there is no path from s to t in $G\}$

$\overline{PATH}$ is co-NL complete under implicitly log-space reduction.

Theorem 12.1. *(Immerman-Szelepcsényi, '88-'89)*

NL = co-NL

Proof Sketch : As $\overline{PATH}$ is co-NL complete, hence using the certificate definition of NL, we know if we give a O(log-space) algorithm A such that for every n-vertex graph G and vertices s and t, there exists a polynomial size read-once certificate u such that $A(G, s, t, u)$ = 1 if and only if t is not reachable from s in G, then we know this language($\overline{PATH}$) belongs to NL. Here, A has only read-once access to u.

Thus, it is sufficient to show that $\overline{PATH} \in NL$

Goal : We need to design a poly-size certificate, u, for an input $x =< G, s, t >$, that will convince the log-space verifier that indeed there is no path from s to t in G. It will follow from the construction that the polynomial certificate can be assumed to be read-once.

Attempt to construct such a certificate :

Suppose G has n vertices and

c_i = Set of all vertices reachable from s by a path of length i or less.

e.g. $c_0 = \{s\}$

c_1 = vertices adjacent to $s \cup \{s\}$ and so on.

Observation : $c_0 \subseteq c_1 \subseteq c_2 \subseteq .... \subseteq c_n$.

Task : Design a certificate that will convince the verifier that t $\notin c_n$.

Question 1 : Suppose the verifier alreadyknows $|c_n|$. Can we design a certificate for "$t \notin c_n$"?

Question 1a : Given a vertex v and an index $i \in [n]$,can we design a certificate for "$v \in c_i$"?

Answer 1a : The certificate is a path from s to v of length i or less.

$B_{0,i,v}$:

s	s_1	s_2	.....	v

This certificate represents that vertex v is reachable from s in atmost i steps. The certificate consists of the path from s to v and every two consecutive vertices are the adjacent vertices in the graph G. This is for every $v \in c_i$ for all $I \in \{0, 1, 2, ..., n\}$.

Question 1b : Is this certificate "read-once"?

Answer 1b : We order the vertices according to some relative ordering on them in G. Hence, this certificate is the "read-once" as we have to check in $B_{0,i,v}$ for (a.) first vertex is s , (b.) every two consecutive vertices in the certificate are adjacent in graph G , (c.) the no. of such pairs is atmost i , and (d.) the last vertex is v. We can check (a.) by the order of s in the relative ordering in G , (b.) by looking into the adjacency matrix of the graph G, (c.) by maintaining a count which should be less than or equal to i, (d.) same as (a.).

Answer 1 : The certificate would be all vertices in $c_n (\neq t)$ and their certificates(paths of length n or less).

$B_{1,n,t}$: v_1 | B_{0,i,v_1} | v_2 | B_{0,i,v_2} | ..... | v_m | B_{0,i,v_m} ,

$v_1 \in c_n$ $v_2 \in c_n$ $v_m \in c_n$

where $|c_n|$ = m , enforce the certificate to be such that $v_1 < v_2 < .... < v_m$.

This certificate represents the certificate for every $v \in c_n$. It contains the certificate for every vertex that is reachable from s in atmost n steps. Hence, we can count all such vertices (and their validity that the certificate is valid) and if the count is equal to $|c_n| = m$ and none of the vertices is t, it means that $t \notin c_n$.

Question 2 : How to convince the verifier that indeed $|c_n|$ = m?

$\rightarrow$ Suppose yes, and the certificate is $B_{2,n}$, then overall certificate(read-once) is :-

$B_{2,n}$	$B_{1,n,t}$

First part of the certificate convinces the verifier that $|c_n|$ = m, while the second part convinces that "$t \notin c_n$"

Question 2a : Suppose the verifier knows that $|c_{n-1}|$. Can we design a certificate to convince that $|c_n|$ = m?

Condition : We know $|c_{n-1}|$

Observation : For every vertex $v \in G$, convince the verifier :

(a) $v \in c_n$: for such vertices we will give the certificate (i.e. $B_{0,n,v}$) as paths from s to v that will contain atmost $n+1$ vertices so that the maximum path length is n. Now we know all the vertices that are reachable from s in atmost i steps. We will check if this equals m because $|c_n| = m$.

or (b) $v \notin c_n$ (see below).

Question 3 : How can we design a certificate that convinces that "$v \notin c_n$" knowing $|c_{n-1}|$ = m'?

u_1	$B_{0,n-1,u_1}$	.....	$u_{m'}$	$B_{0,n-1,u_{m'}}$

i.e. for each u_i, we will check for its validity whether it belongs to c_{n-1}. Hence, we know c_{n-1}. Now, we check that v is not in the $u_{i_{th}}$ row of the adjacency matrix. This ensures that v is not the neighbour of any vertex that belongs to c_{n-1}.Hence, we ensure that $v \notin c_n$. Recursively verify for $|c_{n-1}|$, $|c_{n-2}|$, ... , $|c_0|$ (bottom-up iteratively). $\Rightarrow$ Polynomial-length read-once certificate(it is polynomial as we have total maximum n vertices and for every such vertex, we give the path from s to t (if exists) and we have total of n such certificates $\Rightarrow$ O(n^3) i.e. polynomial in input size). And this is read-once also.

12.2 Polynomial Hierarchy

EXACT INDSET = $\{< G, k > |$ the largest independent set of G has size $k\}$

$< G, k >\in$ EXACT INDSET if $\exists$ an independent set in G of size k and every subset of vertices of size $k+1$ is not independent.

$< G, k >\in EXACTINDSET$ if $\exists u \in \{0,1\}^{p(.)}$ $\forall v \in \{0,1\}^{p(.)}$ $M(x,u,v)$ = 1.

We do not see any short certificate easily for : $< G, k >\in$ EXACT INDSET iff there exists an independent set of size k in G and every other independent set has size atmost k.

MIN-EQ-DNF = $\{< \phi, k > |$ there is a DNF ψ of size k that is equivalent to $\phi\}$.

Here also, we do not see any short certificate of membership for MIN-EQ-DNF. Hence, for such languages such as EXACT INDSET and MIN-EQ-DNF we should not restrict our definition only to "exists" or only "for all" but extend

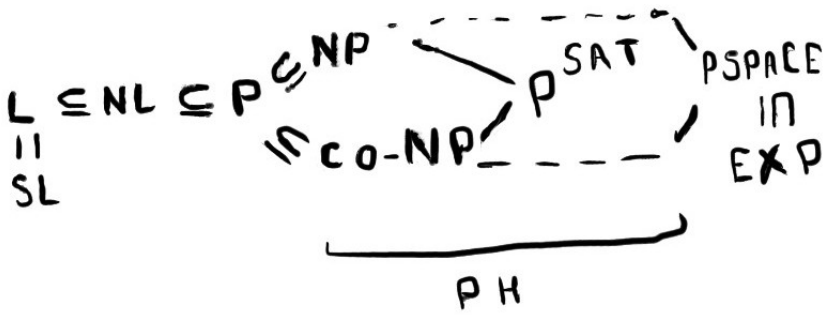

Figure 12.2:

the definition to a mixture of these quantifiers.

Class $\sum_2^P$:

A language $L \in \sum_2^P$ if there is a polynomial function $q(.)$ and a poly-time TM, M such that $x \in L \iff \exists u \in \{0,1\}^{q(|x|)} \forall v \in \{0,1\}^{q(|x|)}, M(x,u,v) = 1$.

$$\sum_1^P = NP \subseteq \sum_2^P$$

EXACT INDSET, MIN-EQ-DNF $\in \sum_2^P$

(for MIN-EQ-DNF, u is the DNF and v is the set of all possible assignments).

Class $\sum_i^P$ (for i≥1) :

A language $L \in \sum_i^P$ if there is a polynomial function $q(.)$ and a poly-time TM, M, such that

$$x \in L \iff \exists u_1 \in \{0,1\}^{q(|x|)}, \forall u_2 \in \{0,1\}^{q(|x|)}, \exists u_3 \in \{0,1\}^{q(|x|)}, ..., Q_i u_i \in \{0,1\}^{q(|x|)} \text{ and}$$

$$M(x, u_1, u_2, ..., u_i) = 1.$$

Observation : $\forall i, \sum_i^P \subseteq \sum_{i+1}^P$.

In figure 12.3, we use Δ_i^P which is defined as $\Delta_i^P = P^{\sum_{i-1}^P}$.

Definition : The Polynomial Hierarchy is the set :

$$\textbf{PH} = \cup_{i \geq 1} \sum_i^P.$$

Definition : $\Pi_i^P = \{L | \overline{L} \in \sum_i^P\}$

$= \text{co-}\sum_i^P$.

Equivalent Definition of PH : PH = $\cup_{i \geq 1} \Pi_i^P$.

Claim : $\sum_i^P \subseteq \Pi_{i+1}^P$.

Proof Sketch : Suppose $L \in \sum_i^P$, then L is defined as

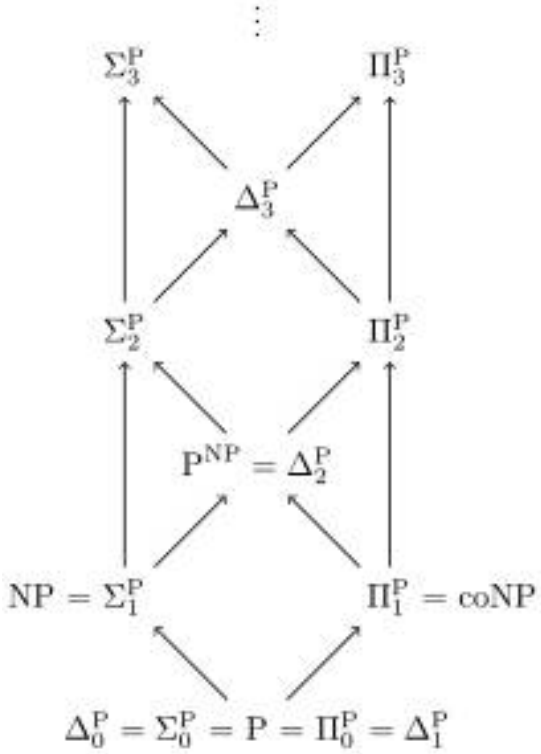

Figure 12.3: Pictorial representation of the polynomial time hierarchy. The arrows denote inclusion (figure is taken from [2]).

$$x \in L \iff \exists u_1, \forall u_2, ..., Q_i u_i, M(x, u_1, u_2, ..., u_i) = 1.$$

So, for such a language, we can design a Turing Machine M' such that with the input we will append a $v \in \{0,1\}^{q(|x|)}$ such that it will not read this v and this v is polynomial in size of x (i.e. $q(.)$ is a polynomial function). Now it can be seen as

$x \in L \iff \forall v, \exists u_1, \forall u_2, ..., Q_i u_i, M'(x, v, u_1, u_2, ..., u_i) = 1$. We can see that this is the definition of $\Pi_{i+1}^P \Rightarrow L \in \Pi_{i+1}^P$.

Here is how $\overline{L}$ looks like :

$$x \notin L \iff \forall u_1, \exists u_2, ..., \neg Q_i u_i, M(x, u_1, u_2, t...., u_i) = 0$$
$$\Rightarrow \quad x \in L \iff \exists u_1, \forall u_2, ..., Q_i u_i, M(x, u_1, u_2, ..., u_i) = 1$$
$$\Rightarrow \quad \Pi_i^P = co\sum_i^P = \{\overline{L} : L \in \sum_i^P\}$$

Equivalent Definition of Π_i^P : Same as $\sum_i^P$ but starts with $\forall$ quantification.

Observation : PH $\subseteq$ PSPACE

Proof Sketch : Pick any language in PH, take PSPACE TM to verify. Any language in PH can be seen as the Quantified Boolean Formula over $\exists$ and $\forall$. So, given a string x and i certificates for some $\sum_i^P$, we can check using the verifier as we can reuse the space used for checking using one certificate.In this way, we have to enumerate for every certificate if quantifier is "for all" and for atleast one certificate for "exists" where it satisfies.Hence, we can see that we have to enumerate over all certificates but at one point we just have to use the space for one certificate.So, total space

required is polynomial in the size of input(i.e. number of quantifiers multiplied by the size of each certificate(i.e. again polynomial in the size of input) $\Rightarrow$ polynomial) Hence, PH $\subseteq$ PSPACE.

12.3 References

[1] S. ARORA and B. BARAK "Computational Complexity: A Mordern Approach," *Cambridge University Press*, 2009

[2] http://en.wikipedia.org/wiki/Polynomial_hierarchy