Recent developments in structural complexity

F. Cucker
J. Díaz

Report LSI–88–20

Abstract: This survey presents some of the recent developments in the field of structural complexity. The survey does not intend to give an exhaustive look in each one of the areas of structural complexity which have been developed recently, but just the ones the authors believe have had a greater impact on the field.

Resum: Aquest article presenta alguns dels avanços recents dins de l'àrea de la complexitat estructural. No es preten donar una visió exhaustiva de totes les àrees de la complexitat estructural que s'han desenvolupat recentment, sino d'aquells resultats dels quals els autors pensen que van tenir un impacte més gran en aquest camp.

Recent Developments in Structural Complexity

F. Cucker[1] and J. Diaz
Llenguatges i Sistemes Informàtics
UPC Pau Gargallo 5, E 08028-BARCELONA

Abstract

This survey presents some of the recent developments in the field of structural complexity. The survey does not pretend to give an exhaustive look into each one of the areas of structural complexity which have been developed recently, but just the ones which the authors believe have had a greater impact on the field.

1 The solution to an old problem

In 1964, Kuroda introduced the nondeterministic linearly bounded automaton and showed that the family of languages accepted by the nondeterministic linearly bounded automata is the same as the family of languages generated by context-sensitive grammars. (For the basic definitions, see [7]). That work raised an important problem; are the languages accepted by deterministic and nondeterministic linearly bounded automata the same?. This problem is known as the *first LBA problem*. If both families of languages are the same, then the family of context-sensitive languages is closed under complementation. On the other hand, it still could happen that both families are different but the family of context-sensitive languages are closed under complementation. In any case, this raised a second open problem, denoted the *second LBA problem*; are context-sensitive languages closed under complementation?. This problems were deeply studied in the work of Hartmanis and Hunt (see [6]) in 1974, where the interested reader will find many relations of these two problems to other problems in complexity theory. Both the first and second LBA problems, remained unsolved, being considered two elements more of the list of unsolved problems in complexity theory. In the summer of 1987, there appeared two independent solutions to the second LBA problem; one by Neil Immerman from Yale University (see[8]), and the other by Róbert Szelepcsényi from Komensky University in Bratislava (see [14]). Eventually, what happened was that while Immerman's report was quickly distributed, at least among the western scientific community, most of the people working in complexity theory did not know anything about Szelepcsényi's result, until a translation of its original article in Slovak, appeared in the Bulletin of the EATCS, in late October 1987 ([15]). By that

[1]Partially suported by DGICyT PB860062

time, Immerman's proof had been deeply studied and discussed, and the initial excitement about the result had cool down a bit. Thus it is not surprising, that our presentation, is based upon Immerman's original paper.

Theorem 1 *For any space constructible function* $s(n) \geq \log n$ *we have:*

$$NSPACE(s(n)) = \ co{-}NSPACE(s(n))$$

Proof.

We wish to show that for any language L, $L \in NSPACE(s)$ if and only if $\overline{L} \in NSPACE(s)$. From the definition, there exists a nondeterministic machine M which accepts any w in L, $|w| = n$, within space $s(n)$. Let us design a nondeterministic machine M', which accepts w using space $s(n)$ if and only if $w \notin L$. The machine M' is described in algorithm 1 below.

First, we shall stablish some conventions. For the machine M and input w, let us consider the instantaneous descriptions (ID) as defined for example in [2]. Each one of these descriptions can be encoded as a chain of length $s(n)$ over the alphabet $\{0, 1\}$. Recall that we don't need to write the input w on each ID, it suffices to write down the position of the input head, which takes at most $\log n$ bits. The machine M' will be based in the following fact; M will reject w if and only if the number of instantaneous descriptions reachable from the initial ID of M on w, I_0 in k or less steps is the same that the number of instantaneous descriptions reachable from I_0 in $k+1$ steps and none of these instantaneous descriptions represents an accepting instantaneous description.

Notice, that the total number of different ID's will be less or equal to $2^{s(n)}$. A key step of the algorithm describing M' will be instruction *get next I*. This instruction, generates in lexicographical order, the chain which follows to the current I, and which repesents a valid ID. The way to do this, is to keep adding in binary 1 to I, until finding the next valid ID. In case there is not such an I, (we are beginning the loop), it starts with the chain formed by $s(n)$ 0's and keeps adding 1 until generating the first chain which represents a valid ID.

The algorithm will use several counters. These can be implemented by laying out $s(n)$ cells in any of the working tapes of the machine. Notice this can be done since $s(n)$ is space constructible. K will count the number of steps we are dealing with at any given moment of the algorithm; D will keep track of the number of reachable configurations at each value of K. As usual, the fact that there exists a path of k or less steps, from configuration I_0 to configuration I, will be denoted by $I_0 \vdash^{\leq k} I$. Counters C_1 and C_2 will contain the number of ID's added in loops 2 and 3 respectively.

Algorithm 1

input(x,M,$s(|x|)$)
create counters D, C_1, C_2, K, and C_3 of size $s(|x|)$
compute initial ID I_0
Comment: Deterministically compute the number C_2 of ID
$C_2 = 0$
get next I
  if deterministically $I_0 \vdash^1 I$, then $C_2 = C_2 + 1$
end get next
$K = 1$
Comment: Begin loop 1, over the number of steps
while $K < 2^{s(|x|)}$
$D = C_2$
Comment: Begin loop 2, over the possible ID which are reachable in at most $k+1$ steps
$C_2 = 0$
1 get next I_2
  Comment: Begin loop 3, over the possible ID from which we can reach I_2 in one step.
  $C_1 = 0$
2 get next I_1
    guess path $I_0 \vdash^{\leq k} I_1$
    if guessed path is correct, then $C_1 = C_1 + 1$
      Comment: Deterministically test if it is possible to go from I_1 to I_2
      if determistically $I_1 \vdash^1 I_2$, then
        if I_2 is accepting, REJECT
        else $C_2 = C_2 + 1$ and go to **1**
    Comment: If not possible to go from I_1 to I_2, go to generate next I_1
    else go to **2**
  end get next
  Comment: If we are at this point, the situation is that we have generated all the possible I_1 which are accessible from I_0, and from none of them we could get I_2
  Comment: We must check that the nondeterminism worked correctly
  if $C_1 \neq D$ then REJECT
 end get next
 if $D = C_2$, then ACCEPT
 else $K = K + 1$
end while

2 Interactive Proof Systems

Recall that the class NP is defined as the class of problems for which there is a known way to check in polynomial time whether a potential solution is an actual solution. Following this remark, let us introduce the following "verifying" machine, which will define the class NP.

Definition 1 *An* NP proof-system *consists of two deterministic Turing machines M_p and M_v, respectively called the* prover, *which has unlimited computing power, and the* verifier, *which is polynomially time bounded. The two machines share a common read-only input tape; they also share a comunication tape, where M_p can only write a chain and M_v can only read the chain written by M_p, and both M_p and M_v have independent reading-writting working tapes.*

The way an NP proof-system defines the class NP is as follows, for any language $A \in NP$, given as input a chain x in A, the prover computes a chain y, with the length of y polynomialy bounded in the length of x, and writes it on the communication tape. M_V reads y and checks in polynomial time that $f_A(x, y) = 1$, where f_A is a polynomial-time computable function which can be thought of a proof of membership of x to A. So in this way, the class NP could be re-defined as the class of sets which have short proofs of membership, where here short means polynomial-time. Let us see some examples to clarify the idea.

Example 1

Let us consider the problem of the clique. Given a graph $G(V, E)$, with $|V| = n$, and a positive integer k, decide whether G contains a complete subgraph of k or more vertices. So A will be the set of chains which codify a graph G and a constant k, such that G contains a clique-k. Our NP proof-system will work as follows; M_p will read the input x which will consist of the graph G properly codified, and the constant k. Recall that x can be codified so $|x| = n \log n$. Let m denote the length of x. Then M_p will compute a clique-k of G, and write it codified as chain y. Then M_v will compute the function f_A , defined by

$$f_A(x, y) = \begin{cases} 1 & \text{if } y \text{ is a subgraph of } x, \text{ it is complete and } |y| \leq k \\ 0 & \text{otherwise} \end{cases}$$

The following procedure will represents this function.

Algorithm 2

Given x and y, to compute $f_A(x, y)$,
recover $G(V, A)$ and k from x
recover the clique $G'(V', A')$ from y
check that G' is indeed a subgraph of G
check that G' is complete

check that $|V'| \geq k$
if the three checks are positive, return 1
else return 0

Example 2

Let us consider now the problem of the graph isomorphism. Given two graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$, with $V_1 = V_2 = \{1, 2, \ldots, n\}$, say that G_1 is isomorphic to G_2 ($G_1 \cong G_2$), if there exists a permutation π from $\{1, 2, \ldots, n\}$ to $\{1, 2, \ldots, n\}$, such that for any vertices i and j, $(i, j) \in E_1$ if and only if $(\pi(i), \pi(j)) \in E_2$. The *graph isomorphism problem* (*ISO*), will be defined by,

$$ISO = \{(G_1, G_2) | G_1 \cong G_2\}$$

Again an *NP* proof-system for *ISO* will consist of the prover reading (G_1, G_2), properly codified, computing the correct permutation π, and send it to the verifier. Then M_V checks in polynomial time that $f_{ISO}((G_1, G_2), \pi) = 1$, where f_{ISO} is the function explicitated in the following algorithm 3,

Algorithm 3
Given input (G_1, G_2) and π
to compute $f_{ISO}((G_1, G_2), \pi)$,
check that for every pair (i, j)
$(i, j) \in E_1$ iff $(\pi(i), \pi(j)) \in G_2$
if true return 1,
else return 0

The above proof-system has an incovenience; it does not allow us any interaction between the prover and the verifier. The *NP*-proof system is like a dumb and inexpresive teacher writting a proof in a class of mute students. In "normal" conditions, we would like to capture a more general way of communicating; for instance, we will like to take advantage of the possibility of students asking questions at key steps of the proof, and receive an answer. This makes teaching easier!. To capture this wider concept of communicating a proof, we define the more general notion of interactive proof system. We do this by stages, defining first the concept of interactive Turing machine, after the concept of interactive protocol and finally the notion of interactive proof sistem.

Definition 2 *An* interactive Turing machine *(ITM) is a Turing machine with a read-only input tape, a read-only random tape, a read-write work-tape, a read-only communication tape, and a write-only output tape. The random tape contains an infinit sequence of bits from* $\{0, 1\}$*, which can be thought of as the outcome of an unbiased coin tosses. When we say that an interactive machine "flips the coin", we mean it reads the next bit from the random tape. The contents of the write-only communication tape can be thought of as messages sent*

by the machine; while the contents of the read-only communication tape can be thought of as messages received by the machine.

An *interactive protocol* is an ordered pair of interactive Turing machines (M_p, M_v), which share the same input tape, the write-only communication tape of M_v is M_p read-only communication tape and vice versa. The machines take turns in being active, with M_v starting the computation. When each machine is active, it can perform internal computation, read and write on the proper tapes and send a message to the other machine by writing on the appropiate communication tape. The i^{th} message of M_p is the entire string that M_p writes on the communication tape during its i^{th} turn. At this point, M_p is deactivated and M_v becomes active, unless the protocol has terminated. The i^{th} message of M_v is similarly defined. Either machine can terminate the protocol by not sending any message in its active stage. The machine M_v *accepts* (*rejects*) the input by entering an accepting (rejecting) state and ending the protocol. M_p is a computationally unbounded Turing machine. The computation time of M_v is defined as the sum of M_v's computation time during its active stages. We require that the computation time of M_v be bounded by a polynomial in the length of the input string. The *number of rounds* of an interactive protocol (M_p, M_v) on input x is defined as the number of messages exchanged between M_p and M_v. The *text of the computation* is the sequence $\{v_1, p_1, \ldots, v_n, p_n\}$, where v_i (p_i) denotes the message send from M_v to M_p (from M_p to M_v) in the i^{th} round. We let $p_n = \lambda$ if M_p halts the computation on the n^{th} round. The *size* of an interactive protocol (M_p, M_v) is defined as the total length of the text of the computation.

As the two interactive Turing machines are random, we can define a probability space on the set of textes of all possible computations of (M_p, M_v) on input x, in such a way that the probability of each computation of (M_p, M_v) on input x is taken over the coin tosses of machines M_p and M_v. This leads us to make the following definitions.

Definition 3 *Given the set $A \in \{0,1\}^*$, we say that A has an* interactive proof-system *if there exists an interactive Turing machine M_v such that,*

1. *There exists an interactive Turing machine M_p such that the pair (M_p, M_v) is an interactive protocol and for every x in A with $|x|$ sufficiently large, the probability that M_v accepts x is greater than $2/3$.*
2. *For every possible interactive Turing machine M_p such that the pair (M_p, M_v) form an interactive protocol, and for every x not in A with $|x|$ sufficiently large, the probability than M_v accepts x is less that $1/3$.*

where as said, the probabilities are taken over the coin tosses of M_p and M_v.
We say that (M_p, M_v) is an interactive proof-system *for A.*

As in done previously in *NP* proof-systems, M_p will be called the *prover*, and M_v will be called the *verifier*.

Interactive proof systems were developed by Goldwasser, Micali and Rackoff in 1985 (see [4]). Their main purpose was to study criptocomplexity questions.

In the previous definition, condition 1 says that for every input in the language A, there exists a way to easily prove this fact to M_v which succeeds with overwhelming probability. Condition 2 says that for every input not in the language, the probability of success of any strategy to fool M_v into believing that x belongs to A, is very small. Other interpretation of this definition is to say that condition 1 means that it is possible to prove a true theorem so that the proofs are "easily" (in polynomial time) verified; and condition 2 means that no one can convince of the proof of a false theorem.

Notice that it does not suffice that the verifier cannot be fooled by the predeterminate prover, as such a condition would have presupossed that the prover is a trusted oracle.

Definition 4 *Given a nondecreasing function q from the naturals to the naturals, we say that (M_p, M_v) is a* q-round interactive proof-system *if for every x in A, the number of rounds in the computation of (M_p, M_v) to recognize x is bounded by q.*

We will consider this fuction to be either bounded by a polynomial of n, and write it $q(n)$, or to be bounded by a constant. Then we can define the following complexity classes;

Definition 5 *Define $IP[q(n)]$ to be the class of languages for which there exists a $q(n)$-round interactive proof-system.*

Notice that the class NP which was discussed at the beginning of the section, is a trivial interactive proof-system where the verifier tosses no coins, in other words $NP = IP[1]$.

From the definition, we can state

Proposition 1

$$NP \subseteq IP[q(n)] \subseteq IP[q(n)+1]$$

where $q(n)$ is a nonconstant polynomial.

Let us see an example of interactive proof system. Define the *graph non-isomorphism* by:

$$NONISO = \{(G_0, G_1) | G_0 \ncong G_1\}$$

that is, $NONISO$ is the set of pairs of graphs, each with set of vertices $\{1, 2, \ldots, n\}$, which are not isomorphics between them.

Given as common input two graphs $G_0(V, E_0)$ and $G_1(V, E_1)$, with $|V| = n$, let us consider the following protocol,

1. M_v reads the input, and it chooses at random n integers $c_i \in \{0, 1\}$. This is done by tossing the coin n times. For each c_i, $1 \leq i \leq n$, M_v computes

a graph $H_i(V, F_i)$ which is isomorphic to $G_{c_i}(V, E_{c_i})$. Notice M_v can do this deterministically within time $n^{O(1)}$. At the end, M_v sends to M_p the n produced graphs $H_1(V, F_1), \ldots, H_n(V, F_n)$.

2. The prover reads the input and tests if G_0 is isomorphic to G_1.(He can do that, as he has unlimited power). In case it is not, the prover reads the message sent by the verifier. For each H_i, M_p tests whether $H_i \cong G_0$ or $H_i \cong G_1$. In the first case he makes $\alpha_i = 0$, in the second he makes $\alpha_i = 1$. If $G_0 \cong G_1$, then for each i, M_p chooses $\alpha_i \in \{0, 1\}$ with probability $1/2$. In any case, when M_p has produced $\{\alpha_1, \ldots \alpha_n\}$, he send them to M_v.

3. The verifier reads the message $\{\alpha_1 \cdots \alpha_n\}$ from the prover. M_v compares each α_i with the corresponding c_i. If for some i he gets that α_i is different than c_i, which means that the prover has made a mistake, the verifier rejects the input. Otherwise, the verifier accepts the input (M_v verifies that $G_0 \ncong G_1$)

Let us prove that this protocol works indeed. In the case that $G_0 \ncong G_1$, then M_p will find that each H_i is isomorphic only to one of G_0 or G_1, but not to both. Therefore M_p will be able to return the correct value of α_i; and in step 3 M_v will find for each i that $\alpha_i = c_i$.

On the other hand, if $G_0 \cong G_1$, then the prover will not be able to decide if M_v has constructed each H_i form G_0 or from G_1, and he will toss the coin to return α_i, with probability $1/2$ of error. Therefore, the probability that M_v does not find at least one i for which $\alpha_i \neq c_i$ is equal to $1/2^n$.

So from the definition we have that *NONISO* belongs to the class *IP[1]*.

It is interesting to note that *NONISO* is not known to be in *NP*. Notice that in the last example the secrecy of the tosses are essential to the protocol. In the next section, we present another proof-system, where the coin tosses are made public.

3 Arthur against Merlin games

The proof-system presented in this section consists in a game where wizard Merlin, which is doted with supranatural intellectual abilities, tries to convince the intelligent human king Arthur, that a string x belongs to a given language L. This proof system, was developed by Babai in 1985 ([1]), and was intended to create as small complexity classes as possible to classify certain algebraic problems, as the isomorphism of matrix groups over finit fields, which are not known to be in *NP*.

In the same way as in the previous section, we define an *Arthur-Merlin protocol* (M, A) on common input x, as an interactive-protocol in which A's messages consists merely of tossing its coin at most a polynomial number of times, an sending their outcomes to Merlin. The protocol is always terminated

by A, who then evaluates a polynomial-time function defined over the contents of its tapes to decide whether to accept or reject the input x. Similarly to the definition of interactive proof-system, we have

Definition 6 *A set L has an* Arthur-Merlin proof-system *if there exists an interactive Turing machine A called Arthur such that,*

1. *There exists an interactive Turing machine M, called Merlin, such that the pair (M, A) is an Arthur-Merlin protocol, and for every x in L with $|x|$ sufficiently large, the probability that A accepts x is greater than $2/3$.*
2. *For every possible interactive Turing machine M such that the pair (M, A) form an interactive protocol, and for every x not in L with $|x|$ sufficiently large, the probability than A accepts x is less than $1/3$.*

As the reader can notice, the difference between the Arthur-Merlin proof-system and the Interactive proof-system is the restricted way that Arthur is allowed to use his coin tosses during the protocol. Thus the Arthur-Merlin proof-system is an special case of interactive proof-systems. Arthur-Merlin proof-systems also give rise to complexity classes.

Definition 7 *Given a nondecreasing function q from the naturals to the naturals we say that (M, A) is a* q-round Arthur-Merlin proof-system *if for every x in L, the number of rounds in the computation of (M, A) that recognizes x is bounded by q.*

Then we can define the following complexity classes;

Definition 8 *Define $AM[q(n)]$ to be the class of languages for which there exists a $q(n)$-round Arthur-Merlin proof-system.*

Babai, proved the following collapse theorem

Theorem 2 *For any constant $k \geq 1$, $AM[k] = AM[k+1]$*

This means that all the Arthur-Merlin games with bounded number of rounds, collapse to $AM[1]$. We shall denote this class by just AM.

From the remarks made above, we have that $AM \subseteq IP$. Moreover, from the example of graph nonisomorphism, it looks like the ability of the verifier to keep the secrecy of its coin tosses is the crucial ingredient which makes the proof work. Surprisingly, it turns out that both classes coincide. This theorem was proved by Goldwasser and Sipser in [5];

Theorem 3 *For any polynomial $p(n)$, $IP[p(n)] = AM[p(n)+2]$*

This theorem is important in the sens that it allow us to use IP to study problems, and AM to look for structural properties. The reason for which the formalism of AM is more suitable to look into structural properties, is that the AM model is a particular case of the general probabilistic classes. A formal global treatment of probabilistic classes, is done in [13].

Definition 9 *Given a class $\mathcal{C}$ of sets, let the* bounded-error probabilistic- $\mathcal{C}$ class *($BP\mathcal{C}$) be defined as the class of all sets A such that for some set B in $\mathcal{C}$, for some $\delta > 0$, for some polynomial p and for all inputs x of length n, we have*

$$\text{Prob}[(x,y) \in B \text{ iff } x \in A] > \frac{1}{2} + \delta$$

where $y \in \{0,1\}^{p(n)}$ is randomly choosen with probability $\frac{1}{2^{p(n)}}$.

When the class $\mathcal{C}$ is P, then we get the well known Bounded error Probabilistic Polynomial time class, BPP. This class has been throughly studied, see for example chapter 6 of [2]. If we take $\mathcal{C}$ as the class NP, we obtain $BPNP$; which is exactly the class AM.

Let us prove that it is possible to "amplify" the probability in definition 11. from $\frac{1}{2} + \delta$ to something like $1 - 2^{-n}$.

An extension of the well known Turing reducibility, is the *positive Turing reducibility* defined by Alan Selman in 1978 ([11]). An oracle machine M is positive iff $A \subseteq B$ implies that $L(M,A) \subseteq L(M,B)$. Then A is positive Turing reducible to B, $A \leq_{pos} B$, iff $A = L(M,B)$ where M is a positive oracle machine. If M is deterministic then we shall write $A \leq^P_{pos} B$. It is a straightforward exercise to prove that the classes NP, P, *co-NP*, $PSPACE$, Σ_i and Π_i for $i \geq 1$, are closed under $\leq^P_{pos}$.

Lemma 1 (Amplification lemma) *Let $\mathcal{C}$ be any class of sets which is closed under $\leq^P_{pos}$. Then for any set $A \in BP\mathcal{C}$ and any polynomial q, there is a set $B \in \mathcal{C}$ and a polynomial p such that for all x with $|x| = n$,*

$$Prob[(x,y) \in B \text{ iff } x \in A] > 1 - \frac{1}{2^{q(n)}}$$

where the y are randomly choosen from $\{0,1\}^{p(n)}$ under a uniform distribution.

Let us introduced a powerful technique that will yield very interesting results. The technique is named the quantifier simulation, and roughly means that if a property holds with high probability, then it can be expressed by an universal or an existential quantifier. This technique was used by Lauteman in 1983 (see [10]), to show $BPP \in \Sigma_2$. The techniques was improved by Zachos and Heller in [16] and we present it as given in [13].

Lemma 2 (Quantifier Simulation Lemma) *Let $E \subseteq \{0,1\}^m$ be some set with cardinality greater than $(1 - 2^{-k})2^m$, where $2^k > m$. Then the following statements are true,*

1. $Prob[\exists v, |v| = m, \forall i \leq m : u_i \oplus v \in E] = 1$
2. $Prob[\forall v, |v| = m, \exists i \leq m : u_i \oplus v \in E] > 1 - 2^{-m(k-1)}$

where $(u_1, \ldots, u_m) \in \{0,1\}^m \times \cdots \{0,1\}^m$ *(m times) is choosen uniformly at random.*

Using the two previous lemmas, we can prove the following important result,

Theorem 4 *For all* $k \geq 1$ *we have,*

$$BP\Sigma_k \subseteq \Pi_{k+1}$$
$$BP\Pi_k \subseteq \Sigma_{k+1}$$

From this last theorem, we obtain the following result due to Babai

Theorem 5

$$AM \subseteq \Pi_2,$$
$$\text{co}-AM \subseteq \Sigma_2$$

Other result about probabilistic classes, which will allow us to derive important consecuences for the class AM, and which again can be deduced from the amplification lemma and the quantifier simulation lemma, is the following

Theorem 6 *For all* $k \geq 1$ *we have,*

if $\Pi_k \subseteq BP\Sigma_k$ *then* $\Sigma_{k+1} = \Pi_{k+1}$
if $\Sigma_k \subseteq BP\Pi_k$ *then* $\Sigma_{k+1} = \Pi_{k+1}$

Rewriting this theorem in terms of AM, we obtain a result from Boppana, Hastad and Zachos ([3]);

Corollary 1 *If either co-*$NP \subseteq AM$ *or* $NP \subseteq$ *co-*AM*, then the polynomial-time hierarchy collapses to* Π_2*.*

Finally, straightforward manipulation of the definition, gives us the following result ([13]),

Theorem 7

$$\Sigma_2(BP\Sigma_k \cap BP\Pi_k) = \Sigma_{k+1}$$

From this theorem a series of important results about AM could be obtained.

Corollary 2

$$\Sigma_2(AM \cap \text{co}-AM) = \Sigma_2$$

Corollary 3 *ISO cannot be NP-complete unless the polynomial time hierarchy collapses to* $\Sigma_2 = \Pi_2$

This last result, due to Boppana, Hastad and Zachos ([3]), and also to Schöning ([12]), is an important one; as it says that unless the polynomial-time hierarchy collapses to its second level (which seems not to be the case), then ISO can not be NP-complete.

4 References

[1] Babai, L., Trading group for randomness. *Proc. 17th ACM Symp. on Theory of Computing*, 1985.

[2] Balcázar, J.L., Díaz, J., Gabarró, J., *Structural Complexity I*, EATCS Monographs in Theor. Comp. Sc., Springer Verlag, 1988.

[3] Boppana, R.V., Hastad, J., Zachos, S. Does co-NP have short interactive proofs?, *Information Processing Letters*, **25**, 1987.

[4] Goldwaser, S., Micali, S., Rackof, C., The knowledge complexity of interactive proof systems. *Proc. 18th. STOC*, 1985.

[5] Goldwasser, S., Sipser, M., Private coins versus public coins in interactive proof systems. *Proc. 18th ACM Symp. on Theory of Computing*, 1986.

[6] Hartmanis, J., Hunt, H.B., The LBA problem and its importance in the theory of computing. In *SIAM-AMS Proceedings* vol. **7**, 1974.

[7] Hopcroft, J., Ullman, J., *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, Reading, Mass., 1979.

[8] Immerman, N., Nondeterministic space is closed under complement. TR, Yale University, July 1987.

[9] Kuroda, S.Y., Classes of languages and Linear-Bounded Automata. *Information and Control*, **7**, 1964.

[10] Lauteman, C., BPP and the polynomial hierarchy. *Information Processing Letters* **14**, 421–429, 1983.

[11] Selman, A., Reductions in NP and p-selective sets. *Theortical Computer Science*, **19**, 287–304, 1982.

[12] Schöning, U., Graph isomorphism is in the low hierarchy. *Proc. 4th. STACS* LNCS 247. Springer-Verlag, 114–124, 1986.

[13] Schöning, U., Probabilistic complexity classes and lowness. *Proc. 2nd. Structure in Complexity meeting*, 1987.

[14] Szelepcsényi, R., Context-sensitive languages are closed under complementation. TR Komensky University, April 1987 (in Slovak).

[15] Szelepcsényi, R., The method of forcing for Nondeterministic Automata. *Bulletin of the EATCS*, october 1987.

[16] Zachos, S., Heller, H., A decisive characterization of BPP. *Information and Control*, **69**, 125–135, 1983.