Recent Highlights in Structural Complexity Theory*

Uwe Schöning

Universität Ulm
Postfach 4066
Germany

Abstract. Recently, several unexpected results about the comparison of certain complexity classes have been obtained. The classes $\oplus P$ and PP have been shown to be "hard" for the polynomial-time hierarchy PH. The class of languages having interactive proofs, IP, was shown to equal $PSPACE$. Furthermore, the GRAPH ISOMORPHISM problem was shown to be "low" for certain complexity classes, and therefore is not likely to be NP-complete. All these results are proved by certain counting techniques and arguments that use probability theory.

Keywords: complexity classes, polynomial hierarchy, interactive proof systems, graph isomorphism

1 Introduction

In the last years, in the area of structural complexity theory, several surprising and counter-intuitive results have been obtained. The purpose of this paper is to review a selection of them, to present at least sketches of the proofs in (what I hope is) an accessible way for readers not familiar with the field.

We found the following results of the last years most remarkable:

Toda [20] has shown that certain counting complexity classes, like PP or $\oplus P$ are much more powerful than expected previously. He shows that all languages in the *polynomial-time hierarchy* are reducible to some language in these classes. Informally, this shows that the concept of *counting* (even counting modulo 2, i.e. computing *parity*) in the context of a nondeterministic, polynomial-time algorithm is at least as powerful as what can be expressed in terms of (polynomially length-bounded) existential and universal quantifications, i.e.

$$x \in A \;\Leftrightarrow\; \exists\, y_1\, \forall\, y_2\, \ldots \overset{\forall}{\exists}\, y_k\; R(x, y_1, y_2, \ldots, y_k)$$

*Invited Lecture at SOFTSEM '91, Nizké Tatry (CSFR), 24.11.91–6.12.91.

where R is a polynomial-time computable relation, and A is the language to be defined.

Toda's proof introduces in a significant way the application and analysis of certain *operators*, applied to complexity classes. We will present this approach by considering operators for existential and universal quantification, for complementation, for the parity operation, and for bounded-error probabilistic computation.

Another recent breakthrough result is due to Shamir [17], who showed that the class of sets being "provable" in terms of an interactive proof system, IP, can capture precisely the class $PSPACE$, so $IP = PSPACE$. Again, the definition of IP can be understood as an application of appropriate operators. Shamir's proof introduces the concept of interpreting quantified Boolean formulas in an arithmetical way and treating them as polynomials over some prime module.

A prominent example, and one of the first known examples, of a problem admitting an interactive proof system, is GRAPH ISOMORPHISM, and also, more surprisingly, GRAPH NONISOMORPHISM [8]. This, in some sense, brings the graph isomorphism problem at least "close" to the class $NP \cap co \cdot NP$. More precisely, it can be shown [15] that GRAPH ISOMORPHISM belongs to the *low hierarchy* in NP [14] and therefore is *not* NP-complete unless the polynomial hierarchy collapses.

That the graph isomorphism problem is "easier" than the NP-complete problems, yet not necessarily polynomial-time solvable, is supported by another very recent result [10]: GRAPH ISOMPORPHISM belongs to a very weak counting class called $LWPP$ [6], and therefore is low for PP.

We will also show that these are not isolated results but that there is a common "toolbox" of techniques that is needed in each of the results. This is showing that Structural Complexity Theory has developed into a well-founded, well-respected field, with its own techniques and methods.

2 Basic Notions from Complexity Theory

Our notation is standard; for unexplained notions see [3].

We will consider several classes of languages, in this context called *complexity classes*, some of which have natural computational interpretations – for example, P is the class of problems considered to have feasible algorithms – others don't have such nice interpretations (especially when they come about using the operators discussed in the next section) but it will be necessary and useful to consider such classes since they serve as intermediate steps in certain proofs.

We mentioned already P, i.e.

$$P = \{A \subseteq \Sigma^* \mid A = L(M) \text{ for some deterministic, polynomial-time Turing machine } M\}$$

In the following, we will also use P for the class of polynomial-time computable *functions*.

The question of whether P is equal to NP where

$$NP = \left\{ A \subseteq \Sigma^* \mid \begin{array}{c} A = L(M) \text{ for some } \textit{non}\text{deterministic,} \\ \text{polynomial-time Turing machine } M \end{array} \right\}$$

is a well-known open problem.

Let $acc_M(x)$ be the number of accepting computation paths of the nondeterministic machine M on input x. The class PP [7], sometimes called $\mathsf{C}P$ for "counting-P", is defined as

$$PP = \left\{ A \subseteq \Sigma^* \mid \begin{array}{l} \text{there is some } f \in P \text{ and a nondeterministic, polynomial-time} \\ \text{Turing machine } M \text{ such that } x \in A \Leftrightarrow acc_M(x) \geq f(x) \end{array} \right\}$$

It is clear that PP includes NP (namely, choose $f(x) = 1$).

For a class of languages $\mathcal{C}$, let $co \cdot \mathcal{C}$ denote the set of complements of the languages in $\mathcal{C}$,

$$co \cdot \mathcal{C} = \{A \subseteq \Sigma^* \mid \Sigma^* - A \ \in \ \mathcal{C}\}$$

This is a first (and simple) example of an operator, which applied to some complexity class, yields a new complexity class. We will discuss an algebra of such operators in more detail in the next section.

It is not known whether NP is closed under complementation, i.e. $NP = co \cdot NP$. But PP is easily seen to be closed under complementation (accepting and non-accepting final states can be interchanged). Therefore, $co \cdot NP \subseteq PP$.

3 Operators on Complexity Classes

Now we introduce several operators acting on complexity classes yielding new complexity classes.

For a class $\mathcal{C}$ denote by $\exists \cdot \mathcal{C}$ the class of sets A for which there is a $B \in \mathcal{C}$ and a polynomial p such that

$$A = \{x \mid \exists y \; [|y| = p(|x|) \wedge B(x, y)]\}$$

For a class $\mathcal{C}$ denote by $\forall \cdot \mathcal{C}$ the class of sets A for which there is a $B \in \mathcal{C}$ and a polynomial p such that

$$A = \{x \mid \forall y \; [|y| = p(|x|) \rightarrow B(x, y)]\}$$

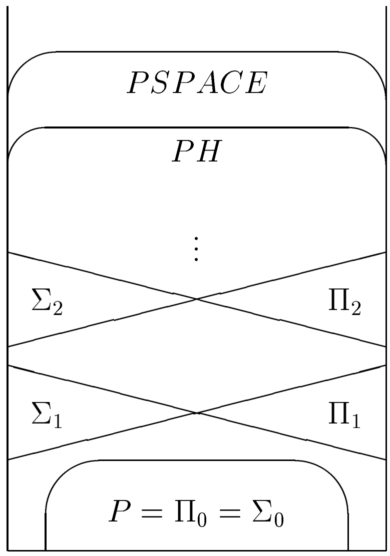
The *polynomial time hierarchy* [19, 22] is the following infinite sequence of classes:

$$\begin{array}{rcl} \Sigma_0 & = & \Pi_0 \ = \ P \\ \Sigma_k & = & \underbrace{\exists \cdot \forall \cdots \overset{\forall}{\exists} \cdot}_{k} P \quad , \; k > 0 \end{array}$$

$$\begin{aligned}
\Pi_k &= \underbrace{\forall \cdot \exists \cdots \overset{\forall}{\exists} \cdot}_{k} P \quad , \ k > 0 \\
PH &= \bigcup_k \Sigma_k \; = \; \bigcup_k \Pi_k
\end{aligned}$$

It is easy to see that PH is included in $PSPACE$, the class of languages recognizable with a polynomially space-bounded machine. It is not known whether the Σ and Π classes form a *proper* infinite hierarchy or whether PH "collapses", i.e., $PH = \Sigma_k$ for some k. The case of $k = 0$ corresponds to $P = NP$. The case of $k = 1$ is equivalent to $NP = co \cdot NP$.

The following diagram sketches the classes of the polynomial-time hierarchy.



Next, we define the parity operator. For a class $\mathcal{C}$ let $\oplus \cdot \mathcal{C}$ be the set of all languages A for which there is a $B \in \mathcal{C}$, a polynomial p, such that

$$A = \{x \mid \text{ there is an odd number of } y,\ |y| = p(|x|), \text{ such that } B(x, y)\}$$

The following operator was introduced in [16]. For a class $\mathcal{C}$ we define $BP{\cdot}\mathcal{C}$ as a probabilistic generalization of $\mathcal{C}$. (BP stands for *bounded-error probabilistic ...*) We let $A \in BP \cdot \mathcal{C}$ if there is a $B \in \mathcal{C}$, $\varepsilon > 0$, and a polynomial p, such that

$$Prob[x \in A \;\Leftrightarrow\; B(x, y)] > \frac{1}{2} + \varepsilon$$

Here, the probability is taken uniformly over all y, $|y| = p(|x|)$.

For classes $\mathcal{C}$ having certain natural closure properties (closure under positive reductions, closure under majority reducibility) it can be shown that the error probability can be made exponentially small, e.g. in the above definition $\frac{1}{2} + \varepsilon$ can be substituted by $1 - 2^{-|x|}$, without changing the defined class (see [16]). This is called *probability amplification*. All classes on which we apply the BP operator in the following enjoy such closure properties.

Some of the classes that can be obtained in terms of these operators have other familiar names in the literature:

Proposition.

$$\begin{array}{rcl} \exists \cdot P &=& NP \;=\; \Sigma_1 \\ \forall \cdot P &=& co \cdot NP \;=\; \Pi_1 \\ BP \cdot P &=& BPP \text{ [11, 18]} \\ BP \cdot \exists \cdot P &=& AM \quad (\text{or } AM(2) \text{ [1]}) \;=\; BP \cdot NP \\ \exists \cdot BP \cdot P &=& MA \quad (\text{or } MA(2) \text{ [1]}) \\ \oplus \cdot P &=& \oplus P \text{ [12, 5]} \end{array}$$

The following inclusion relationships are known: $BPP \subseteq MA \subseteq \Sigma_2 \cap \Pi_2$ [18, 11, 1], $AM \subseteq \Pi_2$ [1], $\oplus P \subseteq PSPACE$.

Now we summarize a collection of relationships between various operators. The notation $\square \to \diamond$ means that the class $\square\mathcal{C}$ is included in $\diamond\mathcal{C}$. Whenever we use $\leftrightarrow$ instead $\to$ equality holds. The reader should keep in mind that some of the mentioned relationships hold only for underlying classes $\mathcal{C}$ with certain closure properties. All classes that we consider here do have these closure properties.

Theorem.

$$\exists \cdot \exists \leftrightarrow \exists \tag{1}$$
$$\forall \cdot \forall \leftrightarrow \forall \tag{2}$$
$$\oplus \cdot \oplus \leftrightarrow \oplus \tag{3}$$
$$BP \cdot BP \leftrightarrow BP \tag{4}$$
$$co \cdot \exists \leftrightarrow \forall \cdot co \tag{5}$$
$$co \cdot \forall \leftrightarrow \exists \cdot co \tag{6}$$
$$co \cdot \oplus \leftrightarrow \oplus \tag{7}$$
$$co \cdot BP \leftrightarrow BP \cdot co \tag{8}$$
$$co \cdot co \leftrightarrow \varepsilon \tag{9}$$
$$BP \to \exists \cdot \forall \tag{10}$$
$$BP \to \forall \cdot \exists \tag{11}$$
$$\oplus \cdot BP \to BP \cdot \oplus \tag{12}$$
$$\exists \cdot BP \to BP \cdot \exists \tag{13}$$
$$\forall \cdot BP \to BP \cdot \forall \tag{14}$$
$$\exists \to BP \cdot \oplus \tag{15}$$

The facts (1)–(4) are straightforward (but they require the class $\mathcal{C}$ to have certain encoding possiblities for pairs of strings.)

Facts (5)–(9) are easily verified where (5),(6) are deMorgan's laws. (7) requires the addition of certain dummy computation paths. In (9), ε is the empty operator.

The first non-trivial facts are (10),(11) which are generalizations of the inclusion $BPP \subseteq \Sigma_2 \cap \Pi_2$ proved in [18, 11]. Here the amplification properties of BP play a vital role.

Straightforward application of probability amplification (requiring the closure properties mentioned above) are the facts (12),(13),(14).

Fact (15) is due to Valiant and Vazirani [21] (in the generalized version introduced by Toda [20]): The idea is that the solution space (i.e. the set of y such that $B(x, y)$ holds) can be hit by a small number of probabilistic restrictions (which randomly cancel out some of the solutions) so that with high probability an *odd* number of solutions remains (in at least one of the restricted solution spaces). If there were no solutions to begin with, then after any such restriction there are still no solutions so there is an even number of solutions. The crucial problem is how to represent such a restriction for an exponentially large space with only polynomially many bits.

4 Toda's result

Toda shows that the class $\oplus P$ is very powerful: Every set in PH can be randomly reduced to some set in $\oplus P$. Formally, this is expressed by the following inclusion relationship.

Theorem [20] $\quad PH \subseteq BP \cdot \oplus \cdot P$

Proof: Let $A \in PH$. Then there is some k such that $A \in \Sigma_k$. Hence we get

$$\begin{aligned}
A \in \Sigma_k \quad &= \quad \underbrace{\exists \cdot \forall \cdots \overset{\forall}{\exists} \cdot}_{k} P \\
&\overset{(9),(6)}{=} \quad \underbrace{\exists \cdot co \cdot \exists \cdot co \cdots \ co \cdot \exists \cdot}_{k} P \\
&\overset{(15)}{\subseteq} \quad \underbrace{BP \cdot \oplus \cdot co \cdot BP \cdot \oplus \cdot co \cdots BP \cdot \oplus \cdot}_{k} P \\
&\overset{(8),(7)}{=} \quad \underbrace{BP \cdot \oplus \cdot BP \cdot \oplus \cdots BP \cdot \oplus \cdot}_{k} P \\
&\overset{(12)}{\subseteq} \quad \underbrace{BP \cdot BP \cdots BP \cdot}_{k} \underbrace{\oplus \cdot \oplus \cdots \oplus \cdot}_{k} P \\
&\overset{(3),(4)}{=} \quad BP \cdot \oplus \cdot P
\end{aligned}$$

□

Corollary. If $\oplus P$ is included in the polynomial hierarchy, then the polynomial hierarchy collapses.

Proof: Suppose $\oplus P \subseteq PH$. Then there is a fixed k such that $\oplus P \subseteq \Sigma_k$. (Since $\oplus P$ has a complete set, such a complete set would be in some Σ_k, and since Σ_k is downward closed under polynomial reductions, all of $\oplus P$ is included in Σ_k.)

Therefore we get

$$\begin{array}{rcl} PH & \subseteq & BP \cdot \oplus P \\ & \subseteq & BP \cdot \Sigma_k \\ & \overset{(11)}{\subseteq} & \forall \cdot \exists \cdot \Sigma_k \\ & \overset{(1)}{=} & \underbrace{\forall \cdot \exists \cdots \overset{\forall}{\exists}}_{k+1} \cdot P \\ & = & \Pi_{k+1} \\ & = & \Sigma_{k+1} \end{array}$$

□

We mention another surprising result by Toda (whose proof builds upon the first result).

Theorem. [20] $PH \subseteq P(PP)$.

(Therefore the class PP cannot be included in the polynomial hierarchy unless the PH collapses.)

5 Shamir's Result

Several years ago, the notion of an interactive proof system was introduced by Goldwasser, Micali and Rackoff [8]. Among other interesting applications (e.g., zero knowledge protocols) the idea can be used to define new (?) complexity classes. Phrased in the framework given in the last sections, a language A can be *proved by polynomially-bounded interactive proof systems within k rounds*, symbolically $IP(k)$, if

$$A \in \underbrace{\exists \cdot BP \cdot \exists \cdot BP \cdots \exists \cdot BP \cdot}_{k} P$$

Such a class can be given an appealing interpretation in terms of two players, called *prover* and *verifier*. Recall that a sequence of alternating $\exists$ and $\forall$ quantifiers can be interpreted as a game: there *exists* a move for player 1 such that *for all* moves of player 2 *there exists* a move for player 1 ... such that player 1 wins (meaning, the predicate becomes true when evaluated with all the moves).

The role of the prover in this type of game here is the same as player 1 above who corresponds to the existential quantifiers. The role of the verifier during the course of the game is only to choose *random* moves. This can be interpreted (from the viewpoint of player 1) as a *game against nature*.

Additionally, a game setting as described is only in accordance with the definition if either there is a strategy for the prover to win with high probability or any prover strategy looses with high probability. The former case corresponds to $x \in A$, the latter case to $x \notin A$.

In an independent paper by Babai [1], the prover and the verifier are called *Merlin* and *Arthur*, resp., and $IP(k)$ is called $AM(k)$.

The polynomial-time hierarchy (which is based on $\exists$ and $\forall$) is believed to be an infinite proper hierarchy. On the other hand, the $IP(k)$ hierarchy (which is based on $\exists$ and BP) definitely collapses:

Theorem. [1, 2] $\bigcup_k IP(k) \;=\; BP \cdot \exists \cdot P \;=\; BP \cdot NP \;=\; AM(2)$

Proof:

$$\begin{aligned} IP(k) \quad &= \quad \underbrace{\exists \cdot BP \cdot \exists \cdot BP \cdots \exists \cdot BP \cdot}_{k} P \\ &\overset{(13)}{\subseteq} \quad \underbrace{BP \cdot BP \cdots BP \cdot}_{k} \underbrace{\exists \cdot \exists \cdots \exists \cdot}_{k} P \\ &\overset{(1),(4)}{=} \quad BP \cdot \exists \cdot P \end{aligned}$$

□

The class $BP \cdot \exists \cdot P$ (or $AM(2)$) is included in level Π_2 of the polynomial hierarchy:

$$\begin{aligned} BP \cdot \exists \cdot P \quad &\overset{(11)}{\subseteq} \quad \forall \cdot \exists \cdot \exists \cdot P \\ &\overset{(1)}{=} \quad \forall \cdot \exists \cdot P \\ &= \quad \Pi_2 \end{aligned}$$

Therefore, $\bigcup_k IP(k)$ is included in Π_2.

Now we abuse our notation a bit, since we allow the number of operators to grow with the input size. This is not quite admissible in the formal sense of our definition but one can easily make the definition formally correct (for example, in terms of a generalization of alternating Turing machines, having existential, universal, parity, and randomizing states).

When we allow polynomially many rounds, we obtain the class $IP(poly)$ (or just called IP.)

$$IP \;=\; \underbrace{\exists \cdot BP \cdot \exists \cdot BP \cdots \exists \cdot BP \cdot}_{poly} P$$

Whether IP equals $IP(k)$ for any fixed k is not known, in fact, in view of the next important result, it seems very unlikely.

Theorem. [17] $IP \;=\; PSPACE.$

Proof: It suffices to show that some $PSPACE$-complete set is in IP. For this purpose, Shamir chooses the set QBF that consists of all valid quantified Boolean formulas. Since the syntax and semantics of such formulas play a crucial role in the proof, we elaborate on this.

The *syntax* of quantified Boolean formulas (qbf) is inductively defined as follows.

> Any variable x_i or its negation $\overline{x_i}$ is a qbf.
>
> If F and G are qbf's, then so are $(F \wedge G)$, $(F \vee G)$.
>
> If F is a qbf and x_i a variable, then $\exists x_i F$ and $\forall x_i F$ are qbf's.

Notice that negation is only allowed on variables.

We define *bound* and *free* variables in the usual way.

We will consider two different semantics for such formulas. The first semantics is the usual Boolean interpretation: The universe is the set $\{0,1\}$. Interpret $\wedge$ as logical *and*, $\vee$ as logical *or*, and $\overline{x_i}$ as logical *negation* of x_i. Interpret $\exists x_i F$ as *there exists a value* $\in \{0,1\}$ *for* x_i *such that* F *gets value 1* and $\forall x_i F$ as *for all values* $\in \{0,1\}$ *for* x_i, F *gets value 1*. We call this the Boolean semantics and denote it by $val_{bool}(F)$.

Next we consider *arithmetical* semantics. The universe is the set of integer numbers, $\mathbb{Z}$. Interpret $\wedge$ as *multiplication*, $\vee$ as *addition*, and $\overline{x_i}$ as *1 minus the value of* x_i. Interpret $\exists x_i F$ as the *sum* of the two values obtained from F when setting x_i to 0 and to 1 and $\forall x_i F$ as the *product* of the two values obtained from F when setting x_i to 0 and to 1. We call this the arithmetical semantics and denote it by $val_{arith}(F)$.

If F is a closed formula, then $val_{bool}(F)$ (resp. $val_{arith}(F)$) evaluates to a constant $\in \{0,1\}$ (resp. $\in \mathbb{Z}$). It is easy to see that in this case,

$$val_{bool}(F) = 0 \;\Leftrightarrow\; val_{arith}(F) = 0$$

Again, the considered $PSPACE$-complete set is

$$QBF = \{F \mid val_{bool}(F) = 1\} = \{F \mid val_{arith}(F) \neq 0\}$$

Notice, if F contains free variables, then $val_{bool}(F)$ resp. $val_{arith}(F)$ becomes a *function* on $\{0,1\}$ resp. on $\mathbb{Z}$. In the arithmetical case such functions can be written as *polynomials*. Such polynomials can be represented by the sequence of their coefficients. Notice that it is computationally hard to find the corresponding polynomial for some given F.

For a given closed formula F (i.e., all variables are bound) define the formula F' such that it is equal to F except that the first occurence of $\exists x_i$ or $\forall x_i$ is taken away. Then, F' has one free variable and (in the arithmetical interpretation) there is a polynomial in one variable that represents F'.

Here is how the (boolean) value of a given qbf F (with k occuring quantifiers) can be determined in terms of $IP(k)$. Here, the p_i are polynomials as discussed above. The following equivalence holds with high probability.

$$val_{bool}(F) = 1 \;\Leftrightarrow\; \exists p_1 \; Rr_1 \ldots \exists p_k \; Rr_k \; B(F, p_1, r_1, \ldots, p_k, r_k)$$

This expression is to be read as follows: The "quantifier" R corresponds to the random choices of the verifier (see the definition of the BP operator). The final predicate B has to be evaluated as follows (letting $\circ_i$ be $+$ if the i-th quantifier in F is $\exists$, and $*$ if it is $\forall$).

$$B(F, p_1, r_1, \ldots, p_k, r_k) \quad \Leftrightarrow \quad \begin{cases} p_1(0) \circ_1 p_1(1) \neq 0 \\ \wedge \; p_1(r_1) = p_2(0) \circ_2 p_2(1) \\ \vdots \\ \wedge \; p_{k-1}(r_{k-1}) = p_k(0) \circ_k p_k(1) \\ \wedge \; val_{arith}(F^*|_{x_1=r_1,\ldots,x_k=r_k}) = p_k(r_k) \end{cases}$$

The intention here is that polynomial p_1 represents F', then the verifier produces a random number r_1 to be assigned to the free variable x_1 in F, hereby obtaining a new (arithmetical) formula F_1, the prover (the existential quantifier) provides a polynomial p_2 that represents F_1' and so on, until there are no more variables in the formula, and a direct evaluation of val_{arith} is possible. Here, F^* is the *matrix* of F, i.e. all quantifiers are taken away, all variables are free. The predicate B checks this final value (where the values $r_1, \ldots, r_k$ are substituted for $x_1, \ldots, x_k$) against $p_k(r_k)$, and checks also that the prover was indeed following the above intention: $p_i(r_i)$ has to be equal to $p_{i+1}(0) \circ_{i+1} p_{i+1}(1)$, and for the qbf to be true, $p_1(0) \circ_1 p_1(1)$ has to be $\neq 0$.

Now, if $val_{bool}(F) = 1$, then the prover can provide the correct polynomials. Therefore the above equivalence holds with probability one.

On the other hand, in the case of the qbf F being false, there is only an exponentially small probability that the verifier can provide the verifier with wrong polynomials (claiming that the arithmetical value is $\neq 0$) and still passes the predicate B.

This shows that $QBF \in IP(k)$ where k is the number of quantifiers in the formula which can grow linearly with the size of the formula. Therefore, $PSPACE \subseteq IP(poly)$. The inverse inclusion $IP(poly) \subseteq PSPACE$ is trivial since a $PSPACE$ machine can traverse the tree of all possible quantifications. Therefore $IP(poly) = PSPACE$. □

Notice that the above proof is just a sketch since we did not mention two details that have to be considered. First, the arithmetical value of a qbf can be double-exponential. Such values cannot be represented with polynomially many bits. But one can see that it is enough to restrict all arithmetical operations to some prime module. The value of such a prime is just exponential, and thus can be represented with polynomially many bits. The prover has to provide such a prime – together with a proof of primality (see [13]) in the first round.

Second, the degree of the polynomials p_i can be exponential in $|F|$ and therefore the coefficients cannot be represented with polynomially many bits. Also this can be fixed.

Shamir defines *simple* qbf's to have a particular syntactic structure which causes the degree to stay linear. He shows that the validity problem for simple qbf's is still $PSPACE$-complete.

A third remark, connected to the previous one, is that simple formulas are not necessarily in prefix form. Therefore, evaluating a polynomial at the points 0 and 1 and multiplying or summing the values, as above, is not adequate. First one has to split F into subformulas $F = F_1 \circ F_2$, $\circ \in \{\wedge, \vee\}$, where F_1 is variable-free, and thus can be directly evaluated, and the polynomial provided by the prover refers to F_2 only (see [17]).

6 Graph Isomorphism

One of the most studied algorithmic problems is graph isomorphism. It is immediate that the language, encoding the graph isomorphism problem, denoted GI, is in NP. But neither membership in P nor NP-completeness has been proved as yet.

We will present several negative results which show that GI is *not* likely to be NP-complete. (This does not imply that GI is in P).

First, the complement of graph isomorphism, $\overline{GI}$ (which is not known to be in NP) has a constant-round interactive proof.

Theorem. [8, 9, 4, 15] $\overline{GI} \in BP \cdot \exists \cdot P$

Proof: Given two graphs G_1, G_2 on n vertices, consider the number of pairs (G, p) such that

> G is an isomorphic graph to either G_1 or G_2,
>
> p is an automorphism of G.

Let $\mathcal{M}$ be the set of such pairs (G, p). Notice that "$(G, p) \in \mathcal{M}$" is a NP predicate.

It can be easily seen [15, 4] that

$$\begin{array}{lcl} G_1 \not\simeq G_2 & \Rightarrow & |\mathcal{M}| = 2n! \\ G_1 \simeq G_2 & \Rightarrow & |\mathcal{M}| = n! \end{array}$$

We specify *random hash functions* mapping the universe of all potential (G, p) to the much smaller space of values $\{1, 2, \ldots, 2n!\}$ in terms of Boolean matrices with random 0,1 entries performing a linear transformation (over $GF(2)$). The number of bits to specify such a matrix is polynomial in n.

Now we have for some constant $\varepsilon > 0$,

$$\begin{array}{lcl} G_1 \not\simeq G_2 & \Rightarrow & Prob_H[\exists\ (G,p) \in \mathcal{M} :\ H(G,p) = 1] > \frac{1}{2} + \varepsilon \\ G_1 \simeq G_2 & \Rightarrow & Prob_H[\exists\ (G,p) \in \mathcal{M} :\ H(G,p) = 1] < \frac{1}{2} - \varepsilon \end{array}$$

Here, a uniform probability distribution over all such hash functions H is used.

Since the predicate "$\exists\ (G,p) \in \mathcal{M} :\ \ H(G,p) = 1$" is in $\exists \cdot P = NP$, this proves that $\overline{GI} \in BP \cdot \exists \cdot P$. $\square$

In the prover-verifier terminology, the above proof can be read as follows.

> First, the verifier picks a random hash funtion H and shows it to the prover.
>
> Then, the prover presents a pair (G,p) – together with a proof of membership of (G,p) in $\mathcal{M}$ – such that H maps (G,p) to the specific point 1.
>
> If such a pair (G,p) can be provided, then the verifier accepts.

The analysis shows, in case of G_1 and G_2 not being isomorphic (i.e. $|\mathcal{M}| = 2n!$), the prover will be able to find such a (G,p) with probability $\frac{1}{2}+\varepsilon$. If G_1 and G_2 are isomorphic (i.e. $|\mathcal{M}| = n!$), the probability of success is ony $\frac{1}{2} - \varepsilon$. By standard methods, the error probability can be further pushed down to 2^{-n}.

Theorem. [4, 15] If GI is NP-complete, then the polynomial hierarchy collapses.

Proof: If GI is NP-complete, then $\overline{GI}$ is $co \cdot NP$-complete. Using the previous theorem, $co \cdot NP = \forall \cdot P \subseteq BP \cdot \exists \cdot P$. Therefore,

$$\begin{array}{rcl} \Sigma_2 & = & \exists \cdot \forall \cdot P \\ & \subseteq & \exists \cdot BP \cdot \exists \cdot P \\ & \overset{(13)}{\subseteq} & BP \cdot \exists \cdot \exists \cdot P \\ & \overset{(1)}{=} & BP \cdot \exists \cdot P \\ & \overset{(11)}{\subseteq} & \forall \cdot \exists \cdot \exists \cdot P \\ & \overset{(1)}{=} & \forall \cdot \exists \cdot P \\ & = & \Pi_2 \end{array}$$

This means that the PH collapses to $\Sigma_2 = \Pi_2$ (even to $BP \cdot \exists \cdot P$). $\square$

In [15] this is further elaborated in terms of *lowness*. It is shown that the graph isomorphism problem is low for the operator Σ_2, that is, $\Sigma_2^{GI} = \Sigma_2$. (Equivalently, the same holds for Π_2). This is shown by proving such lowness result for the whole class $BP \cdot \exists \cdot P$. Roughly, this can be seen by:

$$\forall \cdot \exists \cdot BP \cdot \exists \cdot P \overset{(13)}{\subseteq} \forall \cdot BP \cdot \exists \cdot \exists \cdot P \overset{(1)}{=} \forall \cdot BP \cdot \exists \cdot P \overset{(11)}{\subseteq} \forall \cdot \forall \cdot \exists \cdot \exists \cdot P \overset{(1),(2)}{=} \forall \cdot \exists \cdot P$$

Finally, we want to mention a different type of lowness result, namely, w.r.t. to the counting class PP. First we state the technical theorem, it refers to a class $LWPP$ that was introduced in [6].

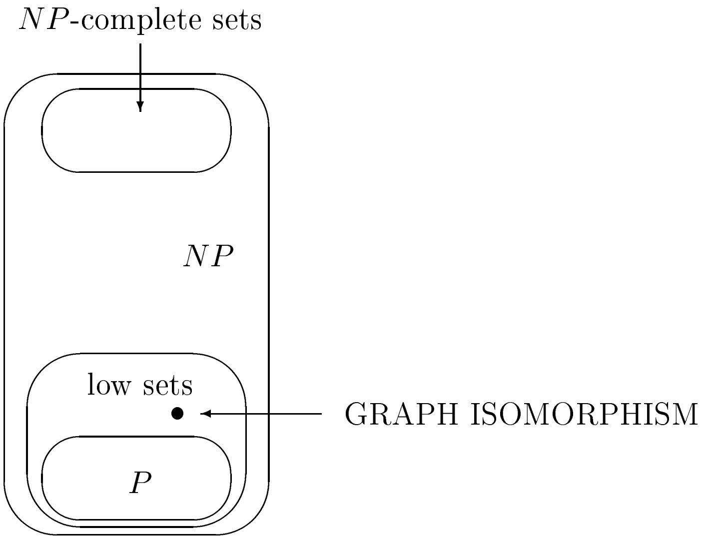
Theorem. [10] $GI \in LWPP$, that is, there is a nondeterministic polynomial-time Turing machine M and two functions $f, g \in P$, $f < g$, such that

$$\begin{aligned} G_1 \simeq G_2 \ &\Rightarrow \ acc_M(G_1, G_2) = g(|(G_1, G_2)|) \\ G_1 \not\simeq G_2 \ &\Rightarrow \ acc_M(G_1, G_2) = f(|(G_1, G_2)|) \end{aligned}$$

Using the results on $LWPP$ proven in [6], we get:

Theorem. [10] GI is low for PP, i.e. $PP^{GI} = PP$. Therefore, if the graph isomorphism problem is NP-complete, then the polynomial hierarchy is low for PP, i.e. $PP^{PH} = PP$.

The following picture gives an impression of the relative position of graph isomorphism in NP.



References

[1] L. Babai. *Trading group theory for randomness.* 17th ACM Symp. Theory of Comput. 1985, pp 421–429.

[2] L. Babai and S. Moran. *Arthur-Merlin games: a randomized proof system and a hierarchy of complexity classes.* Journal of Computer and System Sciences 36, 1988, pp 254–276.

[3] J.L. Balcázar, J. Díaz, J. Gabarró. *Structural Complexity Theory I + II.* Springer-Verlag, 1988 and 1990.

[4] R.B. Boppana, J. Hastad, and S. Zachos. *Does co-NP have short interactive proofs?* Inform. Proc. Letters, 25, 1987, pp 27–32.

[5] J. Cai and L.A. Hemachandra. *On the power of parity polynomial time.* Symp. Theor. Aspects of Comput. Science, Lecture Notes in Computer Science 349, Springer-Verlag, 1989, pp 229–240.

[6] S.A. Fenner, L.J. Fortnow, S.A. Kurtz. *Gap-definable counting classes.* 6th Structure in Complexity Theory Conf., IEEE, 1991, pp 30–42.

[7] J. Gill. *Computational complexity of probabilistic complexity classes.* SIAM Journ. Comput. 6, 1977, pp 675–695.

[8] S. Goldwasser, S. Micali, C. Rackoff. *The knowledge complexity of interactive proof systems.* Proc. 17th ACM Symp. Theory of Computing, 1985, pp 291–304.

[9] S. Goldwasser, M. Sipser. *Private coins versus public coins in interactive proof systems.* Proc. 18th ACM Symp. Theory of Computing, 1986, pp 59–68.

[10] J. Köbler, U. Schöning, J. Torán. *Graph isomorphism is low for PP.* Tech. Report, Fakultt fr Informatik, Universitt Ulm, 1991, submitted for publication.

[11] C. Lautemann. *BPP and the polynomial hierarchy.* Inform. Proc. Letters, 14, 1983, pp 215–217.

[12] C.H. Papadimitriou and S.K. Zachos. *Two remarks on the power of counting.* 6th GI Conf. on Theor. Comput. Science, Lecture Notes in Computer Science 145, pp 269–276, Springer-Verlag, 1983.

[13] V. Pratt. *Every prime has a succinct certificate.* SIAM Journal on Computing 4, 1975, 214–220.

[14] U. Schöning. *A low and a high hierarchy within NP.* Journ. Comput. Syst. Science, 27, 1983, pp 14–28.

[15] U. Schöning. *Graph isomorphism is in the low hierarchy.* J. Comput. System Science, 37, 1988, No. 3, pp 312–323.

[16] U. Schöning. *Probabilistic complexity classes and lowness.* J. Comput. System Science, 39, 1989, pp 84–100.

[17] A. Shamir. *IP = PSPACE.* 31th IEEE Symp. Foundat. Comput. Science 1990, pp 11–15.

[18] M. Sipser. *A complexity theoretic approach to randomness.* Proc. 15th ACM Symp. Theory of Comput. Science 1983, pp 330–335.

[19] L. J. Stockmeyer. *The polynomial-time hierarchy.* Theor. Comput. Science, 3, 1977, pp 1–22.

[20] S. Toda. *On the computational power of PP and* $\oplus P$. 30th IEEE Symp. Found. Comput. Science, 1989, pp 514–519.

[21] L.G. Valiant and V.V. Vazirani. *NP is as easy as detecting unique solutions.* Theor. Comput. Science, 47, 1986, pp 85–93.

[22] C. Wrathall. *Complete sets and the polynomial-time hierarchy.* Theor. Comput. Science, 3, 1977, pp 23–33.