Electronic Colloquium on Computational Complexity, Report No. 71 (2007)

Reductions to Graph Isomorphism

Jacobo Torán
Institut für Theoretische Informatik
Universität Ulm
D-89069 Ulm, Germany
jacobo.toran@uni-ulm.de

August 1, 2007

Keywords: Computational complexity, reducibilities, graph isomorphism.

Abstract

We show that several reducibility notions coincide when applied to the Graph Isomorphism (GI) problem. In particular we show that if a set is many-one logspace reducible to GI, then it is in fact many-one AC^0 reducible to GI. For the case of Turing reducibilities we show that for any $k \geq 0$ an NC^{k+1} reduction to GI can be transformed into an AC^k reduction to the same problem.

1 Introduction

The Graph Isomorphism problem (GI) is one of the few problems in NP that is neither known to be complete for this class nor known to be solvable in polynomial time. Because of its special nature GI has been intensively studied and research on this problem has produced important results in several areas of complexity theory going beyond the GI problem itself. Examples for this are Arthur-Merlin games, interactive proof systems, descriptive complexity or quantum algorithms. The importance of the problem is such, that some authors have used the term GI-complete (see e.g. [5]) for the problems that are equivalent to GI under polynomial time reductions, as if GI were a complexity class. Often computational problems like for example SAT, the

ISSN 1433-8092

set of satisfiable Boolean formulas, or the Graph Reachability problem have been identified with complexity classes. The difference here is that there in no machine model known to characterize the complexity of GI.

In this paper we study several reducibilities to GI proving gap results in the complexity of the models performing the reduction. The results we obtain basically show that the GI problem is very robust under reductions and that in some sense it behaves like a complexity class. We prove that if a set A is reducible to GI under several kinds of reducibility, then the complexity of the reduction can be reduced, and A is in fact AC^0 reducible to GI.

The motivation for studying the complexity of the reductions to GI is twofold. On the one hand, only relatively weak hardness results for GI are known. The strongest known result [10] is that GI is many-one AC^0 hard for DET, the class of problems reducible to the Determinant [4], a class included within NC^2. Several attempts to extend these results to other complexity classes like P, or even NC^2 or AC^1, have not been successful, even under the consideration of reductions that can use more resources than AC^0. The study of reductions to GI give some insight on why it is difficult to improve the known hardness results.

On the other hand our results help to understand the nature of several reducibility notions like for example the AC^k or NC^{k+1} reducibilities. These reducibilities are quite well understood and it is known that both notions coincide when reducing to complexity classes like the NC and AC hierarchies [11], L and NL [2], or NP [8]. We show here that they coincide also when reducing to GI[1]. This is somehow surprising since GI is not a machine based complexity class, and intuitively points to the following property of the reducibilities: If the oracle set is strong enough to encode a logarithmic space computation, then AC^k and NC^{k+1} reducibilities to this set coincide.

Our results are based on a fact that is easy to state: Imagine we have to decide whether two graphs are isomorphic, but the adjacency matrices of these graphs are given in such a way that the 1's and 0's in the matrix are given as pairs of isomorphic and non-isomorphic graphs[2]. How hard is it to decide the isomorphism question then? We show in Lemma 3.2 that this

[1]Ogihara [8] even shows that both reducibilities coincide when performed to a complexity class that is closed under non-deterministic conjunctive truth-table reducibility, but it is not hard to see that the closure of GI under such reducibility is NP and therefore Ogihara's result cannot be applied here.

[2]A more formal version of the statement is given in Lemma 3.2.

problem is not harder that GI itself. This innocent looking fact has many consequences roughly implying that for several reducibilities to GI, part of the complexity of the reduction can be transferred to the isomorphism problem, thus simplifying the reduction. In Section 3 we show that sets many-one NC^1 or logarithmic space reducible to GI are in fact many-one AC^0 reducible to GI. This result can be strengthen to reductions that as strong as the hardest complexity class that can be reduced to GI. Observe that an even stronger gap result is known to hold for SAT. SAT is known to be AC^0 hard for NP (and even NC^0 hard [1]). Since every problem many-one polynomial time reducible to SAT is in NP, it is therefore also many-one AC^0 reducible to SAT. Again, the difference with our result is that we cannot build our proof on a machine based characterization of the complexity class.

In Section 4 we study Turing reducibilities to GI. We show that the classes FL(GI) and AC^0(GI) coincide. Using this fact and adapting a result from [2] on AC and NC reductions to L to the case of GI we prove that for every $k \geq 0$, AC^k(GI) = NC^0(GI).

We conclude with a section of conclusions and open problems.

2 Preliminaries

We assume familiarity with basic notions of complexity theory such as can be found in the standard textbooks in the area.

The elements of the sets we use are encoded as strings over the binary alphabet $\{0,1\}$. A Boolean circuit is an acyclic directed graph with nodes or gates that can either be inputs $x_1,\ldots,x_n$, constants 0 or 1 or are labeled with the AND, OR or NOT functions. Some of the nodes are specified as output nodes $y_1,\ldots,y_m$. A circuit family $\{\alpha_n\}$ computes a function f in the usual way. The size of a circuit is the number of nodes it contains. The depth of a circuit in the length of its longest path from an input node to an output node. The NC and AC hierarchies contain all those functions that are computable by bounded fan-in (resp. unbounded fan-in) circuits of polynomial size and polylogarithmic depth satisfying a certain uniformity condition. Throughout this paper we consider all circuits to be DLOGTIME uniform [9, 3]. Each gate i of a circuit is described by a tuple $\langle i, t, p_1, p_2, ..., p_l\rangle$ specifying the name i of the gate, its type t and the name p_j of its j-th input gate. For $k \geq 0$ we denote by NC^k (resp. AC^k) the class of functions computable by uniform bounded fan-in (resp. unbounded fan-in) circuits of polynomial size and depth $O(\log^k n)$.

L and FL are the classes of set and functions computable by logarithmic space bounded Turing machines.

The known relationships among the considered function classes are:

$$\mathsf{AC}^0 \subseteq \mathsf{NC}^1 \subseteq \mathsf{FL} \subseteq \mathsf{AC}^1 \subseteq \ldots \subseteq \mathsf{NC}^k \subseteq \mathsf{AC}^k \subseteq \mathsf{NC}^{k+1} \ldots$$

2.1 Reducibilities

We deal with many-one and Turing reducibilities. For a function class $\mathcal{F}$ and two sets A and B, we say that A is many-one $\mathcal{F}$ reducible to B ($A \leq_m^{\mathcal{F}} B$) if there is a total function $f \in \mathcal{F}$ such that for every $x \in \{0,1\}^*$, $x \in A \Leftrightarrow f(x) \in B$.

In order to perform Turing reductions, the NC and AC circuits can have access to oracle gates which compute the value of a functional oracle f. For AC circuits, oracle nodes have depth 1. For NC circuits, a oracle gate with m inputs contributes $\log m$ to the depth of the circuit. This is the standard way of counting the depth of oracle nodes [11]. For a complexity class of functions $\mathcal{F}$, we denote by $\mathsf{NC}^k(\mathcal{F})$ and $\mathsf{AC}^k(\mathcal{F})$ the class of functions computable by NC rep. AC circuits of depth $O(\log^k n)$ with oracle access to a function in $\mathcal{F}$. A Turing reduction to an oracle set A can be seen as a reduction to the characteristic function of A.

For the case of FL we will only consider here sets as oracles. $\mathsf{FL}(A)$ is the class of functions that can be computed in logarithmic space making queries to an oracle set A. A closer description of this model is given when it is needed in the proof of Theorem 4.2.

2.2 Graph Isomorphism

An isomorphism between two graphs G and H is a bijection between their sets of vertices which preserves the edges. $G \cong H$ denotes that G and H are isomorphic. GI is the problem

$$\text{GI } = \{(G,H) \mid G \text{ and } H \text{ are isomorphic graphs}\}$$

A central role in some of the proofs will be played by the set of graph pairs $((G,H),(I,J))$ with exactly one of the pairs consisting of isomorphic graphs:

$$\text{PGI } = \{((G,H),(I,J)) \mid G \simeq H \text{ if and only if } I \not\simeq J\}.$$

It is not hard to see that GI is many-one reducible to PGI. But we need a stronger kind of reducibility:

Definition 2.1 Let $\mathcal{F}$ be a class of functions. We say that a set A is strong many-one $\mathcal{F}$ reducible to PGI if there is a total function $f \in \mathcal{F}$ that for every $x \in \{0,1\}^*$ $f(x) = (G,H),(I,J) \in$ PGI and $x \in A \Leftrightarrow G \cong H$.

It is known that every set in NC^1, L, NL and in several other complexity classes is strong many-one AC^0 reducible to PGI [6, 10].

In some of the proofs we will talk about graphs with colored nodes. A color is just a graph gadget or marking that forces the vertices of a color to be mapped to vertices of the same color in every possible isomorphism (see [7]).

For the proof of Lemma 3.2 the following result describing a parity check construction is needed. This result appears implicitly in [10].

Lemma 2.2 *Let $G = (V_G, E_G)$ and $H = (V_H, E_H)$ be two isomorphic graphs containing the two sequences of node pairs $\{u^1_{G_0}, u^1_{G_1}\}, \{u^2_{G_0}, u^2_{G_1}\}, \ldots, \{u^m_{G_0}, u^m_{G_1}\} \subseteq V_G$, and $\{u^1_{H_0}, u^1_{H_1}\}, \{u^2_{H_0}, u^2_{H_1}\}, \ldots, \{u^m_{H_0}, u^m_{H_1}\} \subseteq V_H$ so that any isomorphism between G and H maps pairs in the sequences to the corresponding pairs, i.e. for all i, $1 \le i \le m$, $\{u^i_{G_0}, u^i_{G_1}\}$ is mapped to $\{u^i_{H_0}, u^i_{H_1}\}$. Then from G and H we can construct within AC^0 two new graphs G', H' that are isomorphic if and only if the set of "0 nodes" $u^i_{G_0}$ in G being mapped to "0 nodes" $u^i_{H_0}$ in H is even.*

3 Many-one reducibility

Definition 3.1 Let $A = (V, E)$ be an undirected graph with n vertices. A PGI representation of A is sequence of $\binom{n}{2}$ tuples of PGI graphs (given by their adjacency matrices) $(G^A_{i,j}, H^A_{i,j}), (I^A_{i,j}, J^A_{i,j})$, $1 \le i < j \le n$, such that for every i, j:

$$\begin{aligned} (i,j) \in E &\Rightarrow G^A_{i,j} \cong H^A_{i,j} \text{ and } I^A_{i,j} \not\cong J^A_{i,j}, \\ (i,j) \notin E &\Rightarrow G^A_{i,j} \not\cong H^A_{i,j} \text{ and } I^A_{i,j} \cong J^A_{i,j}. \end{aligned}$$

Our results are based on the following lemma. Intuitively this result can be understood as a version of the fact $\mathsf{NP}(\mathsf{NP} \cap \mathsf{coNP}) = \mathsf{NP}$ scaled down from NP to Graph Isomorphism.

Lemma 3.2 *Consider two undirected graphs A and B with n vertices each, given in PGI representation. There is an* AC^0 *circuit that on input these representations produces the adjacency matrices of two graphs A', B' such that $A \cong B$ if and only if $A' \cong B'$.*

Proof: The idea of the proof is to consider as a basis for A' and B' two cliques K_n^A and K_n^B with n vertices, and substitute each edge (i, j) in the K_n^A-clique by a graph gadget $E_{i,j}^A$ and every edge (k, l)in the K_n^B-clique by a gadget $E_{k,l}^B$ so that $E_{i,j}^A \cong E_{k,l}^B$ if and only if $(G_{i,j}^A \cong H_{i,j}^A$ and $G_{k,l}^B \cong H_{k,l}^B)$ or $(I_{i,j}^A \cong J_{i,j}^A$ and $I_{k,l}^B \cong J_{k,l}^B)$. In other words, $E_{i,j}^A$ and $E_{k,l}^B$ are isomorphic if and only if the edge (i, j) exists in A and edge (k, l) exists in B or both edges do not exist. An isomorphism between A' and B' encodes then a mapping from the vertices of A to the vertices of B (the mapping restricted to the clique nodes) that guarantees that edges in A are being mapped to edges in B and non-edges are being mapped to non-edges. This is an isomorphism between A and B.

Let us define the graph gadgets. For every pair of indices a, b, $1 \leq a < b \leq n$ consider the component $C_{a,b}^A$ containing the four graphs $G_{a,b}^A$, $H_{a,b}^A$, $I_{a,b}^A$, $J_{a,b}^A$s connected in a ring as in Fig. 1. There are six new vertices u^0,u^1, w, x, y and z in the component. A connection in the figure between a graph and one of the new vertices means that there is an edge in $C_{a,b}^A$ between every vertex in the graph and the new vertex.

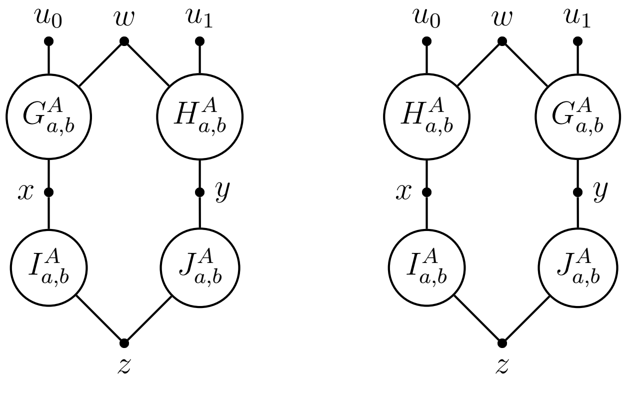

Figure 1: The components $C_{a,b}^A$ and $\overline{C_{a,b}^A}$.

We define also the *twisted* component $\overline{C^A_{a,b}}$ in the same way but interchanging the positions of the graphs $G^A_{a,b}$ and $H^A_{a,b}$. The components $C^B_{a,b}$ are defined in exactly the same way but using the graphs with superscript B. Observe that since we are dealing with PGI graphs, for every a, b, $C_{a,b}$ is isomorphic to $\overline{C_{a,b}}$ (in both cases A and B). Such an isomorphism would map vertex u^0 in $C_{a,b}$ either to u^0 or to u^1 in $\overline{C_{a,b}}$ depending on whether $G_{a,b} \cong H_{a,b}$ or $I_{a,b} \cong J_{a,b}$), exactly one of the two cases is always true.

We are now ready to define the gadgets $E^A_{i,j}$ and $E^B_{i,j}$. Consider i, j with $1 \le i < j \le n$. (For the case $i > j$, $E_{i,j}$ is equal to $E_{j,i}$ for both cases A and B). $E^A_{i,j}$ consists basically of the sequence of components

$$C^A_{1,2}, C^A_{1,3}, \dots, \overline{C^A_{i,j}}, \dots, C^A_{n-1,n}, C^B_{1,2}, \dots, C^B_{n-1,n}.$$

This is the sequences of all the A components followed by all the B components but with the twisted $\overline{C^A_{i,j}}$ component. The components are connected by merging the z vertex of one component and the w vertex of the next component in the sequence. This means that the graph $E^A_{i,j}$ has just one connected component. The gadget $E^B_{i,j}$ is defined in the same way, having all the A components followed by the B components but including the twisted component $\overline{C^B_{i,j}}$ in the sequence instead of $\overline{C^A_{i,j}}$.

Consider now two gadgets $E^A_{i,j}$ and $E^B_{k,l}$ and let us observe that they are isomorphic. An isomorphism between both graphs must map each component in the E^A graph to the same component in the E^B graph. All components are identical except for $C^A_{i,j}$, twisted in the E^A graph and straight in the E^B graph, and $C^B_{k,l}$, straight in the E^A graph and twisted in the E^B graph. We have mentioned that every component is isomorphic to its twisted version and therefore $E^A_{i,j}$ and $E^B_{k,l}$ are always isomorphic. But the type of isomorphism can tell us whether $G^A_{i,j} \cong H^A_{i,j}$ and whether $G^B_{k,l} \cong H^B_{k,l}$. In case $G^A_{i,j} \cong H^A_{i,j}$ the vertex u^0 in $\overline{C^A_{i,j}}$ is mapped to u^0 in $C^A_{i,j}$ and otherwise this vertex is mapped to u^1. Analogously, if $G^B_{k,l} \cong H^B_{k,l}$ then vertex u^0 in $\overline{C^B_{k,l}}$ is mapped to u^0 in $C^B_{k,l}$ and otherwise this vertex is mapped to u^1. Let s be the number of u_0 vertices in $E^A_{i,j}$ being mapped to u_1 vertices in $E^B_{k,l}$. s is either

$$s = \begin{cases} 0 & \text{if } G^A_{i,j} \cong H^A_{i,j} \text{and } G^B_{k,l} \cong H^B_{k,l} \\ 1 & \text{if } G^A_{i,j} \cong H^A_{i,j} \oplus G^B_{k,l} \cong H^B_{k,l} \\ 2 & \text{if } G^A_{i,j} \ncong H^A_{i,j} \text{ and } G^B_{k,l} \ncong H^B_{k,l} \end{cases}$$

This means that the number is even if and only the edges (i, j) in A and (k, l) in B both exist or both do not exist. Since this is the condition we need in order to allow an isomorphism between $E^A_{i,j}$ and $E^B_{k,l}$ we complete the gadget connecting all the u^0 and u^1 vertices in $E^A_{i,j}$ with a parity check construction as in Lemma 2.2 and doing the same thing with the u^0 and u^1 vertices in $E^B_{k,l}$. Finally we mark with a new color the 0-vertices in the output part of the parity constructions of both graphs. This implies that an isomorphism between gadgets $E^A_{i,j}$ and $E^B_{k,l}$ exists if an only if s is even.

Graphs A' results from considering the n-clique K_n and substituting every edge (i,j) by $E^A_{i,j}$. Graph B' is obtained in the same way but substituting edge (i,j) by $E^B_{i,j}$. If every graph in the input tuples $(G_{i,j}, H_{i,j}), (I_{i,j}, J_{i,j})$ has at most m vertices, each gadget $E_{i,j}$ has $\mathrm{O}(mn^2)$ vertices and therefore the size of A' and B' is bounded by $\mathrm{O}(mn^4)$ which is polynomial in the input size. Moreover the construction of A' and B' is completely local and can be performed by an AC^0 circuit. $\Box$

This result can be used to turn part of the complexity of a reduction to GI to the isomorphism problem itself.

Theorem 3.3 *Let L be a set many-one reducible to GI via a function $f : \{0,1\}^* \to \{0,1\}^*$ such that the set*

$$Bit_f = \{\langle x, i, b\rangle \;| x \in \{0,1\}^*, b \in \{0,1\} \text{ and the i-th bit of } f(x) \text{ is } b\}$$

is strongly many-one AC^0 reducible to PGI. Then L is many-one AC^0 reducible to GI.

Proof: If L is many-one reducible to GI then we can consider that for every x $f(x) \in \{0,1\}^*$ is a string representing the adjacency matrices of two graphs A and B, that are isomorphic if and only if $x \in L$. Each bit of $f(x)$ corresponds to one position in one of the adjacency matrices and it is 1 or 0 depending on whether the corresponding edge exists or not. Since the set Bit_f is strongly many-one AC^0 reducible to PGI, there is an AC^0 circuit that produces for each bit of the adjacency matrices two pairs of PGI graphs $(G,H),(I,J)$ with $G \cong H$ if the bit is 1 and $I \cong J$ if it is 0. This is exactly a PGI representation of A and B and by Lemma 3.2 there is an AC^0 circuit that on input this representation produces an adjacency matrix representation of two new graphs A', B' with $A \cong B$ iff $A' \cong B'$. Putting together the strong many-one reduction from Bit_f to PGI and the circuit constructing A' and B'' from the PGI representation of A and B, we have an AC^0 circuit many-one reducing L to GI. $\Box$

This result has several consequences. Basically, if we know that GI is AC^0 hard for a complexity class, then a reduction to GI that uses the resources of this class can be transformed into an AC^0 reduction.

Corollary 3.4 *For any set A, if A is many-one logarithmic space reducible to GI then A is many-one AC^0 reducible to GI.*

Proof: If A is many-one logarithmic space reducible to GI via a function f, then the set Bit_f belongs to L. The result follows from Theorem 3.3 since it is known that every set in L is strongly many-one reducible to PGI [6, 10]. □

Wider gaps in the complexity of the reductions to GI are possible since PGI is known to be hard for complexity classes above L [10]. Although we do not know whether GI is hard for P, the following result relates this question with the equivalence of the closure of GI under many-one reducibilities of different strengths.

Corollary 3.5 *The following statements are equivalent*

i) GI is hard for P under logspace many-one reductions.

ii) the many-one AC^0 and polynomial time closures of GI coincide.

Proof: We show that the first statement implies the second. Let L be a set many-one reducible to GI via a function f computable in polynomial time. The sets

$$\mathrm{Bit}_f^0 = \{\langle x, i\rangle \mid x \in \{0,1\}^*, \text{and the } i\text{-th bit of } f(x) \text{ is } 0\}$$

and Bit_f^1 defined in a similar way are both in P. PGI is strongly many-one AC^0 hard for logarithmic space [10] and therefore, if GI is hard for P under logspace many-one reductions, using Corollary 3.4, GI would be also hard for P under AC^0 reductions. Because of this, both sets Bit_f^0 and Bit_f^1 are many-one AC^0 reducible to GI. Let h_0 and h_1 be the functions performing these reductions. Then, for every $x \in \{0,1\}^*$, $i \in \{1, \ldots, |x|\}$ and $b \in \{0,1\}$, $(h_b(\langle x, i\rangle), h_{\overline{b}}(\langle x, i\rangle))$ are two pairs of PGI graphs and define a strong many-one AC^0 reduction from the set Bit_f to PGI. Now using Theorem 3.3 we conclude that L is in fact many-one AC^0 reducible to GI.

For the other direction, let L be a set in P. L is trivially many-one polynomial time reducible to GI. Since we we are supposing that the many-one polynomial time and AC^0 closures of GI coincide, L is many-one AC^0 reducible to GI and therefore also reducible in logarithmic space to GI. $\square$

Observe that logarithmic space reducibility in the first statement is not really important for the proof of the result. The result would hold also for any reducibility computed by a class of functions with bit sets Bit_f strong many-one AC^0 reducible to PGI.

4 Turing reducibility

Álvarez, Balcázar and Jenner [2] using a functional non-adaptive reduction as an intermediate step, prove the following result:

Theorem 4.1 *[2] For every set A and every $k \geq 0$, $\mathsf{AC}^k(\mathsf{FL}(A)) = \mathsf{NC}^{k+1}(\mathsf{FL}(A))$.*

They prove this result for the oracle function class FL but it can be observed that it relativizes to $\mathsf{FL}(A)$ for any set A queried by the function in FL. In order to apply this result directly to GI (without the FL level) we need the following theorem:

Theorem 4.2 $\mathsf{FL}(\mathrm{GI}) = \mathsf{AC}^0(\mathrm{GI})$.

Proof: Let f be a function in $\mathsf{FL}(\mathrm{GI})$ and M be a logarithmic space bounded Turing machine computing f. A configuration of M contains a state, a position in the input tape and the contents of the work tape. Some of the configurations are query configurations. These contain states of a special kind. If M reaches a query configuration then the machine writes in the following steps a query to GI in the oracle tape and when M enters a special query state the oracle tape is deleted, one bit with the answer to the query appears in it and the computation continues. Observe that the length of the query is not affected by the logarithmic space bound of the work tape. However, the query configuration (of logarithmic size) generating the query, defines the query completely. With this configuration the query can be computed in logarithmic space. Both the number of possible query configurations and the length of $f(x)$ are polynomially bounded in the length of the input x. Consider the set

$$\begin{aligned} A = \{\langle x, K\rangle \quad | \quad & K \text{ is a possible query configuration on input } x \text{ and} \\ & \text{the query produced by this configuration belongs to GI}\}. \end{aligned}$$

A is many-one logarithmic space reducible to GI and as a consequence of Theorem 3.4 also many-one AC^0 reducible to GI. Consider new machine M' that on input a string x and a set of of possible query configurations and answer bits $\langle x, K_1, a_1, K_2, a_2, \ldots, K_m, a_m \rangle$ simulates M on input x and each time M enters a query configuration K, M' looks whether K is part of its input. If this is not the case then it produces some special output sequence and halts. Otherwise M' just continues its computation taking the bit next to K in its input as the answer to the corresponding query. Clearly M' is logarithmic space bounded and computes some function $g \in \mathsf{FL}$. If the set of queries is complete and the set of answers is correct then M' computes f. The set Bit_g is then in L and therefore many-one AC^0 reducible to GI [6]. We want to show that f can be computed in AC^0(GI). In order to do so we just have to put together the AC^0 circuits we already have. On input x the circuit first produces all polynomially many possible query configurations of $M(x)$. Then using the reduction from A to GI, for every such configuration the circuit produces a pair of graphs G, H and queries to the oracle set GI whether they are isomorphic. With the answers the circuit constructs a list of queries and correct answers $x, K_1, a_1, K_2, a_2, \ldots, K_m, a_m$. Finally using the AC^0 circuit reducing Bit_g to GI, for each bit of $f(x)$ a pair of graphs is constructed. A second round of queries to GI gives the value of $f(x)$ in the form of a sequence of bits as output of the circuit.

The constructed circuit has constant depth, polynomial size and has two levels of queries to GI.

□

We can now prove the main result of this section:

Theorem 4.3 *For any $k \geq 0$, $\mathsf{AC}^k(\mathrm{GI}) = \mathsf{NC}^{k+1}(\mathrm{GI})$.*

Proof: The inclusion $\mathsf{AC}^k(\mathrm{GI}) \subseteq \mathsf{NC}^{k+1}(\mathrm{GI})$ is straightforward. For the other inclusion we just have to put together the previous two results. We have $\mathsf{NC}^{k+1}(\mathrm{GI}) \subseteq \mathsf{NC}^{k+1}(\mathrm{FL}(\mathrm{GI}))$ and by Theorem 4.1 this is equal to $\mathsf{AC}^k(\mathrm{FL}(\mathrm{GI}))$. Using Theorem 4.2 this class is equal to $\mathsf{AC}^k(\mathsf{AC}^0(\mathrm{GI}))$. Since every query to $\mathsf{AC}^0(\mathrm{GI})$ can be simulated by the AC^k circuit making the queries directly to GI, just by adding a constant number of levels to the circuit, we have $\mathsf{AC}^k(\mathsf{AC}^0(\mathrm{GI}))=\mathsf{AC}^k(\mathrm{GI})$.

□

We observe that the proofs of Theorems 4.2 and 4.3 can be extended to any complexity class in the oracle that is many one AC^0 hard for L, and for which the many-one AC^0 and logarithmic space closures coincide.

5 Conclusions and open problems

We have proven that several strengths of many-one and Turing reducibilities to GI coincide thus showing that the isomorphism problem is very robust and behaves in some sense as a machine based complexity class. Besides the obvious questions on the complexity of GI there are several problems related to the complexity of reductions that are worth considering:

We know that GI is not hard for NP unless the polynomial time hierarchy collapses. Can one show some relation between the difficulty of showing hardness of GI for a class like P and the hardness for NP? (Something like if GI is P-hard then GI would be NP-hard.)

In this paper we have not talked about randomized reductions to GI. It has been observed in [10] that the Matching problem is randomly reducible to GI. Can also this reduction be simplified making it a deterministic reduction to GI?

We have mentioned that Lemma 3.2 can be considered as a version of the result $\mathsf{NP}(\mathsf{NP} \cap \mathsf{coNP}) = \mathsf{NP}$ scaled down to GI. If the input of the problem given in the lemma instead of being encoded as PGI graphs were just normal graph pairs, isomorphic when encoding a 1 and non-isomorphic when encoding a 0, we would have something like a GI version of the second level of the polynomial time hierarchy. Can one prove a collapse of this hierarchy?

References

[1] M. Agrawal, E. Allender and S. Rudich, Reductions in Circuit Complexity: An Isomorphism Theorem and a Gap Theorem. *J. Comp. Syst. Sci.* 57, 127–143, 1998.

[2] C. Álvarez, J. L. Balcázar and B. Jenner, Adaptive Logspace Reducibilities and Parallel Time. *Math. Systems Theory* 28, 117–140, 1995.

[3] D. A. M. Barrington, N. Immerman, and H. Straubing, On uniformity within NC^1. *Journal of Computer and System Sciences*, 41:274–306, 1990.

[4] S. A. Cook, A taxonomy of problems with fast parallel algorithms. *Information and Control*, 64(1):2–22, 1985.

[5] C. Hoffmann, *Group-Theoretic Algorithms and Graph Isomorphism,* Springer LNCS 136, 1982.

[6] B. Jenner, J. Köbler, P.McKenzie and J. Torán, Completeness results for graph isomorphism. *Journal of Computer and System Sciences*, 66: 549–566, 2003.

[7] J. Köbler, U. Schöning, and J. Torán, *Graph Isomorphism: its Structural Complexity*, Birkhäuser, Boston, 1992.

[8] M. Ogihara, Equivalence of NC^k and AC^{k-1} closures of NP and other classes, Information and Computation, 120,1, 1995, 55–58.

[9] W. Ruzzo, On uniform circuit complexity. *Journal of Computer and System Sciences*, 22:365–383, 1981.

[10] J. Torán, On the hardness of Graph Isomorphism. *SIAM Journal on Computing*, 33, 5: 1093–1108, 2004.

[11] C.B. Wilson, Decomposing NC and AC. *SIAM Journal on Computing*, 19, 2: 384–396, 1990.