Group isomorphism problem and nondeterministic space-bounded classes

Matěj Konečný, Jakub Pekárek, Václav Rozhoň, Štěpán Šimsa

June 2017

1 Introduction

The graph isomorphism problem (GI) is the following problem: one is given two graphs and has to decide whether they are isomorphic or not. Although easy to state, the graph isomorphism problem is one of the few problems known to be in NP for which we do not know whether it is NP-complete or it is in P, although it is strongly believed that the problem is not NP-complete. One reason for this is the recent celebrated quasipolynomial algorithm by Babai [5], the other is a famous result from complexity theory stating that if GI is NP-complete, then the polynomial hierarchy collapses to the second level (PH $= \Sigma_2$) [6]. The key component of the proof is an Arthur-Merlin interactive proof protocol [3, 7] for the graph nonisomorphism problem.

The group isomorphism problem (ΓI) is the problem of deciding whether two groups are isomorphic. The problem can be reduced by polynomial reduction to GI [8]. This means that also for ΓI one can show that if it is NP-complete, the polynomial hierarchy collapses to the second level. It is believed, though, that ΓI is strictly easier than GI. One reason for this is that $\Gamma\mathrm{I} \subseteq \mathrm{NSC}_2$, a class that is believed to be substantially smaller than NP.

We study the group isomorphism problem from the complexity-theory point of view. The tools we use are mainly the ones that were used to prove the above-mentioned complexity-theoretical result on GI and for brevity we usually do not give full proofs, mainly when they go along similar lines as the appropriate proofs of results concerning graph isomorphism (for details, see, e.g. [2] or [4]).

Besides ΓI we also study the quantifier hierarchy defined over space bounded classes $\mathrm{SC}_k = \mathrm{TIMESPACE}(poly(n), \log^k(n))$. The classes of our interest are $\mathrm{SC}_1 = L$ (which was already extensively studied and seems to be well-understood), and SC_2 that we study with respect to the group isomorphism problem. We use the notation NSC_k for the nondeterministic variant of SC_k.

The starting point of this enterprise is the following unpublished result due to Papakonstantinou.

Theorem 1.1. ΓI *is in* NSC_2.

Sketch of proof. Each group can be generated by $\mathcal{O}(\log(n))$ of its elements. Guess the generators of both the first and the second group (we need $\mathcal{O}(\log^2(n))$ bits to store them), close the both sets under the inverse operation. Further guess the isomorphism of these two generating sets. With a help of a deterministic algorithm for finding paths (Reingold's algorithm) we can deterministically express each element of a group as a product of the generators (path in the corresponding Cayley graph). Then we can easily check whether the guessed function is really an isomorphism. □

For believing that a problem is easy, it is valuable to know that not only the problem itself lies in a class of relatively easy problems, but also its complement does. To this end we define space bounded analogues of so-called interactive proof systems in Section 2. This is done in a non-standard way as the standard approach would give classes that are too weak for our methods. We believe that our approach may be of its own interest, as it closely relates to the standard polynomial hierarchy. In the subsequent Sections 3 and 4 we then place the group non-isomorphism problem in classes defined by space-bounded interactive proofs with public and private coins, respectively.

2 Different models

While the following results are stated for the Σ hierarchy (starting with existential quantifier), one can extend the results to Π hierarchy in the straightforward manner.

Definition 2.1. *(Strong and weak hierarchy) For class of problems $C = \mathrm{TIMESPACE}(f, g)$ we define two types of hierarchies. Weak hierarchy is the hierarchy defined by alternations and we will denote its levels Σ_k^C and Π_k^C. Strong hierarchy is defined in a similar manner and we use symbols $\overline{\Sigma_k^C}$ and $\overline{\Pi_k^C}$ to denote its levels.*

We say that a multi-tape Turing machine M is nondeterministic-read-only, if it has one two-way multiple-read input tape, one one-way write-only output tape and some number of one-way read-only nondeterministic (or random) tapes (i.e. it reads the bits in the given order and each can be read only once). The nondeterministic tapes are not space-bounded.

Both hierarchies (where either $S = \Sigma_k^C$ or $S = \overline{\Sigma_k^C}$) can be defined as sets of languages such that $L \in S$ if and only if there exists a nondeterministic-read-only machine M with resources in C and

$$x \in L \Longleftrightarrow \exists a_1 \forall a_2, \ldots, Q a_k : M(x, a_1, a_2, \ldots, a_k) = 1.$$

The difference between the weak and strong hierarchies is how the machine M gets the nondeterministic bits. In the weak hierarchy they are all on one tape written in the same order as is the order of quantifiers[1]*, while in the strong hierarchy there is one tape for every a_i.*

The following observations can be seen immeadiately from the definition:

Observation 2.1. *For any $C \subseteq C'$ we have $\Sigma_k^C \subseteq \Sigma_k^{C'}$ and $\overline{\Sigma_k^C} \subseteq \overline{\Sigma_k^{C'}}$.*
Moreover, $\Sigma_1^C = \overline{\Sigma_1^C}$ and $\Sigma_k^C \subseteq \overline{\Sigma_k^C}$.

We can also easily observe that the two notions are the same for $C = \mathrm{TIMESPACE}(poly, poly)$, because the machine can start by copying the content of its non-deterministic tape(s) on the worktape.

Observation 2.2.
$$\Sigma_k^{\mathrm{P}} = \overline{\Sigma_k^{\mathrm{P}}}.$$

In the following paragraphs we are going to be using the interactive proof classes MA and AM. For precise definitions cf. TODO. Here, by $(\exists^+ x)\varphi(x)$ we mean that for at least $\frac{2}{3}$-fraction of feasible choices of x, the formula $\varphi(x)$ is satisfied. Using this notation, one can say that a language L is in MA if there is a polytime computable formula $\varphi(x, a, r)$ and $x \in L \leftrightarrow (\exists a)(\exists^+ r)\varphi(x, a, r)$ and $x \notin L \leftrightarrow (\forall a)(\exists^+ r)\neg\varphi(x, a, r)$, for AM the order of the quantifiers is reversed.

The classes MA and AM can be also defined in both the weak and the strong sense in a completely analogous manner to the SC_k hierarchy. We will use $\overline{\mathrm{MA}_{\mathrm{SC}_k}}$ is in $\overline{\mathrm{AM}_{\mathrm{SC}_k}}$ for the strong classes.

Note that due to the space restriction we are not able to amplify the probability of success in the weak notion of AM and MA by grouping more than constant number of rounds of the same protocol in one interaction.

Because of this, we choose to work with the strong notion with multiple tapes. In this notion we can amplify probabilistic protocols by their repeated application and by an argument completely analogous to the one that $\mathrm{MA}_P \in \mathrm{AM}_P$ one can prove that $\overline{\mathrm{MA}_{\mathrm{SC}_k}}$ is in $\overline{\mathrm{AM}_{\mathrm{SC}_k}}$, as switching the order of quantifiers does not alter the arrangement of the tapes.

However, this notion of hierarchy is very strong, as $\mathrm{SAT} \in \overline{\mathrm{MA}_L}$:

Proposition 2.3. $\mathrm{NP} \subseteq \overline{\mathrm{MA}_L}$.

[1]this is the usual definition

Proof. (sketch) We solve SAT by a machine from $\overline{\mathrm{MA}_L}$. Note that any problem from NP is reducible to SAT by L-reductions.

Nondeterministic tape of our machine contains an assignment of variables of the formula. We uniformly randomly choose one of m clauses from the formula and verify that the clause is satisfied by the non-deterministic assignment. If the assignment is not satisfactory, then there is at least on un-satisfied clause, and a probability at least $1/m$ of choosing it and hence answering correctly. We amplify the probability by repeating this protocol. $\square$

Similarly, one can easily observe that NP $\subseteq \overline{\Sigma_2^L}$. Note that although the classical property of the weak hierarchy is that for every $C = \mathrm{TIMESPACE}(f, g)$ it holds that $\Sigma_1^C = \Pi_1^C$ implies $\bigcup_{i=1}^n \Sigma_i^C = \Sigma_1^C$, this does not seem to hold in the strong notion of hierarchy. This is because for an L machine we actually know that that the premise is true (NL = coNL), but we will soon see that the consequence of an analogous result would mean that PH = NL. Similarly, derandomization of $\overline{\mathrm{MA}_L}$ (i.e., proof $\overline{\mathrm{MA}_L}$ = NL) implies that NP = NL.

We follow by clasifying the complete problems for both the weak and the strong notion of hierarchy in the interesting case when the underlying complexity class C is equal to $\mathrm{SC}_k = \mathrm{TIMESPACE}(\mathit{poly}(n), \log^k(n))$.

2.1 Complete problems for the hierarchy

The complete problem for $\overline{\Sigma_k^{\mathrm{SC}_m}}$ is similar to the complete problem for the polynomial hierarchy. Its corresponding language contains formulas with their pathwith bounded by $\log^m(n)$ satisfying:

$$\varphi \in L \iff (\exists a_1)(\forall a_2)\dots(Qa_k)\varphi(a_1, a_2, \dots, a_k) = 1.$$

For the proof of the case with $k = 1$, as well as the definition of pathwith for formulas see [1, 9]. Note that the formula has to be given together with its path decomposition. The general case can be easily proved in the same way.

The complete problem for $\Sigma_k^{\mathrm{SC}_m}$ is the same as the complete problem for $\overline{\Sigma_k^{\mathrm{SC}_m}}$ with the additional restriction that in the pathwidth decomposition there are at first the bags with variables from a_1, then variables from a_2, etc. (variables from a_i and a_{i+1} can intersect in one bag). The proof is straightforward and similar to the previous case.

2.2 Strong hieararchy and PH

In this section we prove that there is a strong relation between the strong hierarchy over L and the standard PH hierarchy. Specifically, we prove the following theorem:

Theorem 2.4. *For every $k \geq 2$:*

$$\Sigma_{k-1}^{\mathrm{P}} \subseteq \overline{\Sigma_k^{\mathrm{L}}}$$

First recall that any problem in Σ_k^{P} is L-reducible to Σ_k^{P}SAT (the k-quantifier satisfiability problem). Now we define an auxiliary model of the hierarchy $\widetilde{\Sigma_k^{\mathrm{L}}}$ that is defined as $\overline{\Sigma_k^{\mathrm{L}}}$ except that we do not restrict the machine to read the nondeterministic tapes only once.

Lemma 2.5. $\Sigma_k^{\mathrm{P}}\mathrm{SAT} \subseteq \widetilde{\Sigma_k^{\mathrm{L}}}$.

Proof. We propose a $\widetilde{\Sigma_k^{\mathrm{L}}}$ machine solving Σ_k^{P}SAT. Suppose the formula is already in the CNF form (otherwise we construct an equivalent CNF formula in an online manner. We interpret the nondeterministic tapes as the assignments of corresponding variables, take the clauses one by one and for each we check whether the assignment is correct (we scan the nondeterministic tape once each time we need a nondeterministic variable). $\square$

Now we are ready to prove Theorem 2.4. Although quite technical when written down, the idea behind the proof is very easy. We want to prove the inclusion $\widetilde{\Sigma^{\mathrm{L}}_{k-1}} \subseteq \overline{\Sigma^{\mathrm{L}}_{k}}$. This we will simply do by repeating the nondeterministic bits on each tape enough times so that we can simulate multiple reads. Then, of course, one has to check whether all the copies are the same. To do that for, say, $(\forall a)\varphi(x,a)$, one can devise new equi-satisfiable formula

$$(\forall a)(\exists i)\left(\text{some copies in } a \text{ differ on the } i\text{-th bit} \vee \varphi(x,a)\right).$$

If the copies differ, then the bits on tape a are not part of what we want to check, so we simply accept. Otherwise we can simulate multiple reads by having enough copies of the nondeterministic bits. For the existential quantifier, one universally (over index i) checks whether all the bits on position i are the same and the formula is satisfied at the same time.

The formal proof follows:

Proof of Theorem 2.4. By Lemma 2.5 we know that $\Sigma^{\mathrm{P}}_{k-1} \subseteq \widetilde{\Sigma^{\mathrm{L}}_{k-1}}$, so it is enough to show that $\widetilde{\Sigma^{\mathrm{L}}_{k-1}} \subseteq \overline{\Sigma^{\mathrm{L}}_{k}}$. Let M be a machine working in $\widetilde{\Sigma^{\mathrm{L}}_{k-1}}$ and $L(M)$ the corresponding language. We will construct a machine M' in $\overline{\Sigma^{\mathrm{L}}_{k}}$ such that $L(M') = L(M)$.

Denote the nondeterministic tapes of M resp. M' as $t_1, \dots, t_{k-1}$ resp. $t'_1, \dots, t'_k$ (in the order of the corresponding quantified variables). We know there exists a polynomial $P(n)$ such that M does at most $P(n)$ steps for $|x| = n$. We will treat every tape as having several parts. First of them will be long $\lceil \log(P(N)) \rceil = O(\log(n))$ bits and all the other parts will be long $P(n)$ bits and there will be $P(n)$ of these parts. The first tape does not have the first part and the last tape, t'_k, has only the first part. We will start by reading the first parts and storing them to our work tape (so for every tape $2 \le i \le k$ we have some number b_i). Now M' simulates M step-by-step, but after each step it jumps to the next part of each nondeerministic tape.

Whenever we read part of some tape i we remember its b_{i+1}-th bit and check if it is equal to the b_{i+1}-th bit of the previous part of same the tape (for any tape we remember at most two bits for every tape by throwing away old bits).

If we found two bits that were not equal on the first (existential) tape, we reject. Otherwise, if we found two bits that were not equal on the second (universal) tape, we accept. Generally, if there was no difference on tapes $t'_1, \dots, t'_{l-1}$ and there is a difference on tape t'_l we accept if l as an existential tape and reject otherwise.

If we did not find any not matching bit on any of the tapes we accept/reject base on our simulation of M.

We now prove that $L(M) = L(M')$.

We know that

$$x \in L(M) \iff (\exists a_1)(\forall a_2)\dots M(x, a_1, a_2, \dots, a_{k-1}) = 1. \qquad (1)$$

We have M' and the content of the first tape is a'_1, the content of the second tape is $b_2 a'_2$, third tape $b_3 a'_3$, and so on until the last tape contains only b_k.

First suppose that $x \in L(M)$. We want to show that $x \in L(M')$. We will choose a'_1 (the content of first tape of M') as $a_1^{P(n)}$ where a_1 is what we get from Equation 1 ($a_1^{P(n)}$ means $P(n)$ copies of a_1 padded with zeros to length $P(n)$). That means that no matter what b_2 is we will not find a difference between two parts on the first tape and so we will not reject because of this.

Then there are two possibilities for a'_2. Either $a'_2 = c^{P(n)}$ for some c or not. If not then we can choose b_3 such that we find a difference between two different parts of a'_3 and then we accept (as we wanted). Otherwise we take $a_2 = c$ and Equation 1 gives us a_3. Now we continue with choosing $a'_3 = a_3^{P(n)}$ and proceed the same as with a'_1.

When we constructed a'_{k-1} and b_k we either accepted because of some difference in a'_{2l} or all the tapes $a'_1, \dots, a'_{k-1}$ contain $P(n)$ exact copies of $a_1, \dots, a_{k-1}$ for which M accepts. But then our machine M' simulates the machine M with the access to the same inputs and because M accepted, M' will accept as well.

The second part of the proof $x \notin L(M) \implies x \notin L(M')$ can be done analogically. $\square$

Trivially, the second inclusion would be true for any class $\mathrm{L} \subseteq C \subseteq \mathrm{P}$ and then $\overline{\Sigma_k^{\mathrm{L}}} \subseteq \overline{\Sigma_k^{C}}$ so the Theorem 2.4 is true for any such class C.

Theorem 2.6. *For every $k \geq 1$:*

$$\overline{\Sigma_k^{\mathrm{L}}} \subseteq \Sigma_{k-1}^{\mathrm{P}}$$

Proof. For $k = 1$ we get the well-known theorem $\mathrm{NL} \subseteq \mathrm{P}$. For general k the proof goes along the same lines. State of the machine at each step of computation can be described by the content of work tape, position of heads reading input/work/nondeterministic tapes and the transition state of the machine. All of this information can be described by $\mathcal{O}(\log(n))$ bits.

Now when simulating a $\overline{\Sigma_k^{\mathrm{L}}}$ machine M by $\Sigma_{k-1}^{\mathrm{P}}$ machine M' we at first guess the assignment of the first $k-1$ tapes of M on the $k-1$ corresponding tapes of M'. Then we build the state graph of the machine M of size $poly(n)$ (the same as in the proof $\mathrm{NL} \subseteq \mathrm{P}$). Because the content of the first $k-1$ tapes is fixed now the only states with outdegree two in the state graph are those that correspond to reading the last nondeterministic tape while all of the other states have outdegree one. Now the existence of the content such that M accepts is clearly equivalent to existence of an appropriate path in the state graph that can be found in polynomial time. □

3 $\Gamma\mathrm{NI}$ is in $\mathrm{IP}_{\mathrm{SC}_2}$

In this section we give an interactive protocol for group nonisomorphism for machine in SC_2. The proof goes along the same lines as the classical proof of graph nonisomorphism from ?? (choose one group at random, permute its elements and send it to the prover asking him what group we have sent). However, as the memory of the verifier is limited, it cannot store the whole information of the chosen permutation of the group. We overcome this problem by not sampling uniformly from the set of all permutations, but rather we choose only from a subset of permutations that are generated from the partial permutation of the generating sets of size $\mathcal{O}(\log n)$. This set of generators can be stored in the limited memory of an SC_2 machine.

The reason why we can use the notion of generating sets is the following simple lemma (it can be viewed as the special case of the Alon-Roichmann theorem).

Lemma 3.1. *Let Γ be a group with at least two elements. Then for each ε there is a c_ε such that at most ε fraction of subsets of Γ of size $c_\varepsilon \log(n)$ is not generating.*

Sketch of proof. One by one, we will randomly pick elements into the set $A \subseteq \Gamma$. As the set generated by A is a subgroup of Γ, either $\langle A\rangle = \Gamma$ and we have a generator, or by Lagrange's theorem $|\langle A\rangle| \leq |\Gamma|/2$, hence the probability, that in the next step we pick an element outside of $\langle A\rangle$ is at least $\frac{1}{2}$ and thus enlarge the size of the generated subgroup at least by a factor of two (again by Lagrange's theorem). Then it is enoug to pick a suitable c_ε to get the probability of failure below ε. □

As we have already stated, the crucial idea is to sample a permutation of group elements only from the limited set induced by a partial permutation of a small generating set. The heart of the argument is then to show that if our two groups are isomorphic, then the resulting distribution of the generated Cayley tables for the first group is the same as the one for the second group.

Definition 3.1. *For a permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ we define $\widetilde{\pi}$ to be the corresponding permutation matrix.*

Theorem 3.2. *There is a machine $M \in \mathrm{SC}_2$ such that*

- *M takes as an input a Cayley table C_Γ and an ordered set of indices $(s_i)_{i=1}^k$;*
- *M outputs NO if elements $(s_i)_{i=1}^k$ do not generate Γ;*
- *otherwise M outputs $\bar{C}_\Gamma = \widetilde{\pi} C_\Gamma$ for some permutation π, where the elements of $\bar{C}_\Gamma$ are canonically renumbered $0, 1, 2, \ldots$ in this order and the elements of s_i form an initial segment of the table following the order s_i; and*

- $\widetilde{\pi}M(C_\Gamma, (s_i)_{i=1}^k) = M(\widetilde{\pi}C_\Gamma, (\pi s_i)_{i=1}^k)$ *for every permutation* π *of group elements.*

Moreover, for π *an injective function from the generating set* $(s_i)_{i=1}^k$ *to* Γ *it holds that* $M(C_\Gamma, (s_i)_{i=1}^k) = M(C_\Gamma, (\pi s_i)_{i=1}^k)$ *if and only if* π *(uniquely) extends to an automorphism of* Γ*.*

To propose such an algorithm we use a Cayley graph representation of Γ. As a subroutine we use Reingold's algorithm. We will need that this algorithm satisfies a natural property that its output does not depend on the underlying representation of the graph it uses (i.e. on the order of the vertices).

Definition 3.2. *Let* G *be a* k*-regular graph. We define* $S(G)$ *as a representation of* G *via an ordered set of ordered lists of neighbors of each vertex, i.e.* $S(G)_i = (v_1, v_2, \ldots, v_k)$*, where vertices* $\{i, v_j\} \in E(G)$ *for every* $j \in \{1, \ldots, k\}$*.*

Definition 3.3. *For permutation* $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ *we define* $\pi(S(G))$ *as permutation of vertices of* G *while preserving the order of edges incident to every vertex. So for every* $i \in \{1, \ldots, n\}$*:*

$$\pi(S(G))_{\pi(i)} = (\pi(v_1), \pi(v_2), \ldots, \pi(v_k)),$$

where $S(G)_i = (v_1, \ldots, v_k)$*.*

Proposition 3.3 (s, t connectivity in small space)**.** *There is a machine* $M_{st} \in \mathrm{L}$ *such that*

- M_{st} *takes as an input* $S(G)$ *and two vertices* s *and* t *of* G*,*
- M_{st} *outputs list of indices from* $S(G)$ *defining path from* s *to* t *or* $false$ *if no such path exists,*
- $\pi(M_{st}(S(G), s, t)) = M_{st}(\pi(S(G)), \pi(s), \pi(t))$ *for every permutation* π *of vertices of* G

Proof. One can check that the Reingold's algorithm has all the desired properties. $\square$

Now we use this algorithm to prove Theorem 3.2.

Sketch of proof of Theorem 3.2. First we close the $(s_i)_{i=1}^k$ under taking inverse and add the unit element, such that the closure order the original elements of s_i go first, then the unit element (if not present in s_i already) and then all the missing inverses, again following the order s_i. We get an l-tuple of logarithmic size. Now we can define a Cayley graph G from this l-tuple and represent it as $S(G)$, where the order of neighbours of each vertex is the same as the order in this l-tuple. We can use machine M_{st} from Proposition 3.3 to produce a path from every element in G to 1. If one of these paths does not exist than the graph is not connected and that is equivalent to the fact that the original k-tuple was not a generating set. In this case we return NO. Otherwise, for every element g in the group we define its position in the Cayley table we output as follows – if $g = s_i$ then it is i and otherwise it is $l + x$ where x is the number of elements among $\Gamma \setminus \{s_i\}$ that are smaller than g with respect to lexicographical order of the paths from 1 to g. To compare two elements with respect to this order we run two instances of M_{st} simultaneously. We return the Cayley table C_Γ element by element. To find the index stored at position (i, j) we find the two elements with indices i and j in our lexicograhical order, multiply them (we use the Cayley table from input tape) and return the index of the new element in our order.

Now the fourth property of the theorem follows from Proposition 3.3. The last property follows from the fact that the permutation of generators has a unique extension to the whole group. If the equality $M(C_\Gamma, (s_i)_{i=1}^k) = M(C_\Gamma, (\pi s_i)_{i=1}^k)$ holds, then the permutation of generators uniquely extends to an automorphism of Γ. On the other hand, no automorphism other than identity is identical on any set of generators. $\square$

Now we can easily build the sampling procedure with the desired property that if the two groups are isomorphic, then the distribution of groups we sample does not depend on the group we choose to sample from.

Theorem 3.4. ΓNI *has an interactive protocol with private bits and verifier from* SC_2*.*

Idea. Pick $p \in \{0,1\}$ uniformly at random. Pick $c \cdot \log(n)$ different group element indices $s_i, s_2, \ldots, s_{c \log n}$ at random. Generate C_{Γ_p} permuted according to (s_i) one element at a time using the algorithm from 3.2 and send it to the prover. Prover replies with $q \in \{0,1\}$ indicating which group on the input is isomorphic to the group defined by C_{Γ_p}. Accept if and only if $p = q$. This can be easily amplified in such a way that the probability of success can be any constant (combine several rounds of interaction in one). □

4 ΓNI is in $\mathrm{AM}_{\mathrm{SC}_2}$

Definition 4.1. *($\mathrm{AM}_{\mathrm{SC}_2}$ model) Input tape (random access), random tape (read once), non-deterministic tape (read once), memory $log^2(n)$, run-time $poly(n)$.*

As in the corresponding proof of GI we use the so-called lower bound method, where we choose a suitable set representing symmetries of our two groups and then argue that the set is substantially larger if the two groups are non-isomorphic. As in the case of the IP protocol from the previous section we choose only a suitable subset of symmetries that can be described in small space – in this we differ from the corresponding proof for GI where one works with the set of all permutations of the given graph.

The suitable set S is defined as a union of S_1 and S_2 where for $i \in \{1,2\}$ we have $S_i = \{(\widetilde{\Gamma}, \pi) \mid \widetilde{\Gamma} = M(C_{\Gamma_i}, s_1, \ldots, s_k), (s_1, \ldots, s_k) \in \Gamma_i^k, \pi \in \mathrm{Aut}(\widetilde{\Gamma})\}$ where M is the machine from Theorem 3.2.

Observe that $|S_i| = |\{s_1, \ldots, s_k \in \Gamma_i : s_1, \ldots, s_k \text{ are disjoint and generating}\}|$. This immediately follows from the last part of Theorem 3.2, as we know that $M(C_{\Gamma_i}, s_1, \ldots, s_k) = M(C_{\Gamma_i}, t_1, \ldots, t_k)$ iff the partial function mapping each s_i to t_i extends to an automorphism π of Γ_i. On the other hand, any automorphism of Γ also specifies uniquely a partial automorphism of its first k elements.

From Lemma 3.1 we can easily infer that there is a suitable c such that after defining $k = c \log n$ we have

$$0.9 \cdot n(n-1)\cdots(n-k+1) \leq |\{s_1, \cdots, s_k \in \Gamma_i, \text{ disjoint and generating}\}| \leq n(n-1)\cdots(n-k+1).$$

In other words, most of the sets are generating.

Further observe that if $\Gamma_1 \simeq \Gamma_2$ then $|S| \leq n(n-1)\ldots(n-k+1)$. If $\Gamma_1 \not\simeq \Gamma_2$ then $|S| \geq 1.98 \cdot n(n-1)\ldots(n-k+1)$. Thus we are in the right situation for the set lower bound protocol. Note that any member of the disjoint union $S_1 \sqcup S_2$ can be represented by the sequence $s_1, \ldots, s_k$, π restricted to this set and one bit b stating whether the sequence refers to the element of the first or the second group. Thus, any member of S can be represented by $\mathcal{O}(\log^2 n)$ bits. For successful application of the set lower bound protocol we first need to verify that given such a representation $(s_1, \ldots, s_k), \pi, b$ we can verify that this is, indeed, a member of S. Then we need to find an appropriate 2-universal family of hash functions such that for given y we can check whether $h(s_1, \ldots, s_k, \pi, b) = y$. The latter problem is subject to the following proposition.

Proposition 4.1. *There is a 2-universal family of hash functions $\mathcal{H}$ from $\{0,1\}^{p(n)}$ to $\{0,1\}^k$ with $p(n)$ being a polynomial in n and $k = \mathcal{O}(\log^2(n))$ such that:*

- *Computation of $h(x), h \in \mathcal{H}$ is in SC_2 if we have random access to bits of x.*
- *The random bits required to choose h from $\mathcal{H}$ can be read only once.*

Sketch. Define $\mathcal{H}$ as the set of functions $h_{b_1,\ldots,b_k,\beta}(x)$ for all $b_i \in \{0,1\}^{p(n)}, \beta \in \{0,1\}^k$, where the result of each $h_{b_1,\ldots,b_k,\beta}$ is defined as a concatenation of k bits, each computed by scalar product $< b_i | x >$ for $x \neq 0$ or the $h_{b_1,\ldots,b_k,\beta}(x) = \beta$ for $x = 0$. Note that we can choose our function $h \in \mathcal{H}$ just by reading the strings $b_1, b_2, \ldots$ and compute each dot product in a streaming fashion, thus we do not need to store any bit from the b_i's. □

Theorem 4.2. $\Gamma\mathrm{NI} \in \mathrm{AM}_{\mathrm{SC}_2}$

Sketch. This is done by following the classical lower bound protocol.

- Arthur chooses y from a suitable range.
- Merlin sends a message containing $x = ((s_1, \ldots, s_k), \pi, b)$, where π is a permutation of $\{1, \ldots k\}$ and b is a bit representing which group to sample from.
- Arthur first verifies that x is valid by checking that

$$M(C_{\Gamma_b}, (s_i)_{i=1}^k) = M(C_{\Gamma_b}, (\pi(s_i))_{i=1}^k),$$

  where M is the machine from Theorem 3.2. Then he checks that $h\left((M(C_{\Gamma_b}, (s_i)_{i=1}^k), \pi)\right) = y$ following Proposition 4.1.

The protocol can be repeated to give an exponentially small probability of failure. $\square$

References

[1] Eric Allender, Shiteng Chen, Tiancheng Lou, Periklis A. Papakonstantinou, and Bangsheng Tang. Width-parametrized sat: Time–space tradeoffs. *Theory of Computing*, 10(12):297–339, 2014.

[2] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach.* Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[3] Laszlo Babai. Trading group theory for randomness. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, STOC '85, pages 421–429, New York, NY, USA, 1985. ACM.

[4] Laszlo Babai. E-mail and the unexpected power of interaction. *Structure in Complexity Theory Conference, 1990*, 1990.

[5] Laszlo Babai. Graph isomorphism in quasipolynomial time. *arXiv:1512.03547*, 2015.

[6] Ravi B. Boppana, Johan Hastad, and Stathis Zachos. Does co-np have short interactive proofs? *Information Processing Letters*, 25(2):127 – 132, 1987.

[7] S Goldwasser and M Sipser. Private coins versus public coins in interactive proof systems. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, STOC '86, pages 59–68, New York, NY, USA, 1986. ACM.

[8] Gary L. Miller. Graph isomorphism, general remarks. *Journal of Computer and System Sciences*, 18(2):128 – 142, 1979.

[9] Periklis A. Papakonstantinou. A note on width-parameterized sat: An exact machine-model characterization. *Information Processing Letters*, 110(1):8 – 12, 2009.