Completeness results for Graph Isomorphism [1]

Birgit Jenner [a], Johannes Köbler [b], Pierre McKenzie [c], Jacobo Torán [a]

[a] *Abt. Theoretische Informatik, Universität Ulm, D-89069 Ulm*

[b] *Institut für Informatik, Humboldt Universität zu Berlin, D-10099 Berlin*

[c] *Université de Montréal, CP 6128 succ Centre-Ville, Montréal QC H3C 3J7*

Abstract

We prove that the graph isomorphism problem restricted to trees and to colored graphs with color multiplicities 2 and 3 is many-one complete for several complexity classes within $\mathbf{NC}^2$. In particular we show that tree isomorphism, when trees are encoded as strings, is $\mathbf{NC}^1$-hard under $\mathbf{AC}^0$-reductions. $\mathbf{NC}^1$-completeness thus follows from Buss's $\mathbf{NC}^1$ upper bound. By contrast, we prove that testing isomorphism of two trees encoded as pointer lists is $\mathbf{L}$-complete. Concerning colored graphs we show that the isomorphism problem for graphs with color multiplicities 2 and 3 is complete for symmetric logarithmic space $\mathbf{SL}$ under many-one reductions. This result improves the existing upper bounds for the problem. We also show that the graph automorphism problem for colored graphs with color classes of size 2 is equivalent to deciding whether a graph has more than a single connected component and we prove that for color classes of size 3 the graph automorphism problem is contained in $\mathbf{SL}$.

1 Introduction

The graph isomorphism problem GI consists in deciding whether there is a bijection between the nodes of two given graphs, G and H, preserving the edge relation. GI is one of the most intensively studied problems in Theoretical Computer Science. Besides its many applications, the main source of

Email addresses: `jenner@informatik.uni-ulm.de` (Birgit Jenner), `koebler@informatik.hu-berlin.de` (Johannes Köbler), `mckenzie@iro.umontreal.ca` (Pierre McKenzie), `toran@informatik.uni-ulm.de` (Jacobo Torán).

[1] Results included in this paper have appeared in the conferences CCC'98 [1] and STACS'02 [2].

interest in GI has been the evidence that this problem is probably neither in **P** nor **NP**-complete. Other sources of interest include the sophistication of the tools developed to attack the problem (for example [3,4]), and the connections between GI and structural complexity (see [5]).

Understandably, many GI restrictions have been considered. For example, **P** upper bounds are known in the cases of planar graphs [6] or graphs of bounded valence [4]. In some cases, like trees [7,8] or graphs with colored vertices and bounded color classes [9], even **NC** algorithms for isomorphism exist ($\mathbf{NC}^k$ is defined to be the class of problems solvable by a uniform family of bounded in-degree Boolean circuits with $O(\log^k n)$ depth and polynomial size, and $\mathbf{NC} = \cup_k \mathbf{NC}^k$).

However, until recently none of these GI restrictions were known to be complete for a natural complexity class, and it seemed that these problems lack the structure needed for a hardness result. Torán proved against this intuition in [10] that GI for general graphs is hard for **NL** (nondeterministic logarithmic space) and for every logarithmic space class based on counting. We consider here GI restricted to trees and colored graphs, presenting the first completeness results for the isomorphism problem restricted to a family of graphs. We obtain completeness results for the complexity classes $\mathbf{NC}^1$, **L** and **SL**.

In the case of trees, a linear sequential time algorithm for tree isomorphism TI was already known in 1974 to Aho, Hopcroft and Ullman [11]. In 1991, an **NC** algorithm was developed by Miller and Reif [12]. One year later, Lindell [7] obtained an **L** upper bound. Finally, in 1997, a subtle algorithm able to test two trees for isomorphism in $\mathbf{NC}^1$ was devised by Buss [8].

Buss in [8] asks whether tree isomorphism is $\mathbf{NC}^1$-hard. Here we answer this question affirmatively showing the hardness of the problem under $\mathbf{AC}^0$ many-one reducibility. Hence tree isomorphism is $\mathbf{NC}^1$-complete. Trees thus provide the first class of graphs for which the isomorphism problem captures a natural complexity class. Moreover, so far, the problem of evaluating a Boolean formula [13] and the problem of multiplying permutations on five points [14] (and some of their variations) were the only two $\mathbf{NC}^1$-complete problems known. Tree isomorphism is a third such problem.

As noted by Buss, choosing a graph representation is critical when working at the level of $\mathbf{NC}^1$. Buss uses Miller and Reif's *string* representation for trees on the grounds that, when the *pointer* representation is used, the "deterministic transitive closure problem can be reduced to the descendant predicate, and the former is known to be complete for logspace" [8, Section 2].

We prove here that the tree isomorphism problem is **L**-hard under many-one $\mathbf{AC}^0$-reducibility when the pointer representation is used. Hence, tree isomorphism in the pointer representation is **L**-complete by Lindell [7] or by

Buss [8]. Tree isomorphism thus captures in this way another very natural complexity class.

The other graph family we consider is the class of colored graphs. We will denote by b-GI the isomorphism problem for colored graphs with color multiplicities bounded by b. Isomorphisms between colored graphs are color preserving and this reduces the number of possible isomorphisms, but observe that even for the case $b = 2$ the number of color-preserving maps between n-vertex graphs could be as large as $2^{\frac{n}{2}}$. This problem also has a fairly long tradition in the area of algorithms. Babai gave in [15] a random polynomial-time algorithm for testing isomorphism when the color multiplicities are less than a constant b. The algorithm was improved to be deterministic in [16] and using sophisticated algebraic tools, Luks proved in [9] that b-GI lies in fact in **NC**.

We focus here on the family of colored graphs with color multiplicities bounded by the constants 2 and 3 proving that 2-GI and 3-GI are many-one complete for symmetric logarithmic space **SL** under $\mathbf{AC}^0$ reductions. The complexity class **SL** introduced in [17] has different characterizations, but the easiest way to define it is precisely as the class of problems logarithmic space reducible to the reachability problem for undirected graphs, UGAP. Our results improve on the one hand the upper bound for the problem given by Luks from **NC** to **SL** (**SL** is included in **NL** which in turn is included in $\mathbf{NC}^2$). On the other hand the results provide new natural examples for complete problems in **SL** showing that reachability questions can be expressed in a natural way as isomorphisms in a class of graphs.

Closely related to graph isomorphism is the graph automorphism problem GA that consists in deciding whether a given graph has a non-trivial automorphism, or in other words, whether there is a permutation of the nodes, different from the identity, preserving the adjacency relation. The relationship between GA and GI is not completely clear. It is known that GA is many-one reducible to GI [18] but a reduction in the other direction is not known and GA seems to be an easier problem. We observe that this situation also occurs within the restricted graph families considered in this paper. We show that testing whether a tree in the pointer representation has a non-trivial automorphism is complete for **L**, while for the string representation the automorphism problem does not seem to have enough structure to be complete for $\mathbf{NC}^1$. We also prove that 2-GA and 3-GA belong to **SL** and moreover 2-GA is equivalent to the problem $\overline{\text{UCC}}$ of deciding whether a given graph has more than one connected component. $\overline{\text{UCC}}$ belongs to **SL** but it seems easier than UGAP and it is not known to be complete for **SL** [19].

2 Preliminaries

We assume familiarity with basic notions of complexity theory such as can be found in the standard books in the area. In particular, we simply recall that

$$\mathbf{AC}^0 \subseteq \mathbf{NC}^1 \subseteq \mathbf{L} \subseteq \mathbf{SL} \subseteq \mathbf{NL} \subseteq \mathbf{NC}^2,$$

where $\mathbf{AC}^0$ is the class of languages recognized by **DLOGTIME**-uniform families of constant depth, polynomial size, Boolean circuits of unbounded fan-in over the basis $\{\wedge, \vee, \neg\}$, $\mathbf{NC}^k$ is the class of languages recognized by **DLOGTIME**-uniform families of polynomial size and depth $\mathrm{O}(\log^k)$, Boolean circuits of bounded fan-in over the basis $\{\wedge, \vee, \neg\}$, **L** is the set of languages accepted by deterministic Turing machines using logarithmic space, **SL** is the set of languages accepted by nondeterministic symmetric Turing machines logarithmic space and **NL** is the set of languages accepted by nondeterministic Turing machines using logarithmic space.

2.1 Isomorphism and Automorphism

Given two graphs $G = (V_1, E_1)$ and $H = (V_2, E_2)$ an isomorphism between G and H is a bijection $\varphi : V_1 \longrightarrow V_2$ satisfying for every pair of nodes $u, v \in V_1$

$$(u, v) \in E_1 \Longleftrightarrow (\varphi(u), \varphi(v)) \in E_2.$$

We will denote by $\mathrm{Iso}(G, H)$ the set of isomorphisms between both graphs. The Graph Isomorphism problem, GI, consist in deciding whether there exists some isomorphism between two given graphs.

We denote by $Aut(G)$ the set of automorphisms in G, that is the set of bijections from V_1 onto itself satisfying

$$(u, v) \in E_1 \Longleftrightarrow (\varphi(u), \varphi(v)) \in E_1.$$

GA is the problem of deciding whether a given graph has a non-trivial (different from the identity) automorphism.

2.2 Reducibilities

We state our results using **DLOGTIME**-uniform many-one $\mathbf{AC}^0$ reducibility [20]. For languages $A, B \subseteq \Sigma^*$, we say that A is many-one $\mathbf{AC}^0$ reducible to B if there is a function f computed by a DLOGTIME-uniform family of $\mathbf{AC}^0$ circuits having the property that, for all $x \in \Sigma^*$, $x \in A$ iff $f(x) \in B$. We write in this case $A \leq_m^{\mathbf{AC}^0} B$.

For simplicity in the case of general graphs our isomorphism and automorphism results are stated for undirected graphs. It is a well known fact (see e.g. [5]) that deciding isomorphism for directed or undirected graphs are equivalent problems. For the case of trees however the situation is not so clear since the standard transformation of an undirected graph into a directed one (considering two directed edges, one in each direction, for each undirected one) introduces cycles. A consequence of our results is that the equivalence under many-one $\mathbf{AC}^0$ reductions holds also for undirected and directed tree isomorphism. Except when the contrary is explicitly stated, we will consider that the trees discussed in this paper are rooted (hence implicitly directed) and unordered (i.e. the ordering of the descendants of a node does not matter). As mentioned, we will argue that the results also hold for unrooted trees, or for rooted trees in which the direction of the edges is explicitly given.

In some of our constructions we use colored graphs. A graph with n nodes is said to be *colored* if each node in the graph is labeled with a positive integer not greater than n. An *isomorphism* between two colored graphs preserves the colors and preserves the edges.

We consider two different representations for encoding trees: the string representation and the pointer representation. As we will see, in the small complexity classes we are dealing with, the representation used might change the complexity of the problem.

In the *string representation* [12], trees are represented over an alphabet containing opening and closing parentheses. A string representation of a tree T is defined recursively in the following way: The tree with a single node is represented by the string "()", and if T is a tree consisting of a root and subtrees $T_1, \ldots, T_k$ (in any order), with string representations $\alpha_1, \ldots, \alpha_k$, a representation of T is given by "$(\alpha_1, \ldots, \alpha_k)$". Observe that a tree might have different string representations, depending on the order of the descendants of any of its nodes. Colored trees can be encoded in the same way using colored parentheses. Let C be the set of colors. An opening parenthesis in a colored tree is represented by "(" followed by $\log(|C|)$ bits encoding the color. Note that we only need to color the opening parentheses.

The *pointer representation* is a more standard way to encode graphs. In this case we consider the trees given by a list of pairs of nodes representing directed edges. As in the previous case, if we deal with colored trees, with the representation of a node we include $\log(|C|)$ bits to encode its color.

For many tree problems in $\mathbf{NC}^1$ and $\mathbf{L}$, completeness results seem to depend on the representation used. For example, the reachability problem on forests, which is **L**-complete in the pointer representation [21], can be solved in $\mathbf{NC}^1$ (and even $\mathbf{TC}^0$) in the string representation [13]. Analogously, the Boolean formula value problem, which is complete for $\mathbf{NC}^1$ in the string representation [8], becomes **L**-complete when described using trees given in the pointer representation [22]. In fact, changing from pointer to string representation is **FL**-complete [22].

Another important observation is that it is possible to test within the classes $\mathbf{NC}^1$ and **L** whether a given input is a correct encoding of a tree in the string representation, and respectively, in the pointer form.

3 Tree Isomorphism for Trees Given by Strings

We show first that deciding whether two trees given in string representation are isomorphic is $\mathbf{NC}^1$-complete. We first consider colored trees for which the hardness proof is simpler. We denote by CTI the isomorphism problem for colored trees.

Lemma 3.1 *In the string representation,* CTI *is* $\mathbf{NC}^1$*-hard under* $\leq_m^{\mathbf{AC}^0}$*-reducibility.*

Proof. Barrington, Immerman and Straubing show in [20] that an $\mathbf{NC}^1$ circuit can be simulated by a balanced **DLOGTIME**-uniform family of Boolean expressions made up of alternating layers of ANDs and ORs. Because of this fact, it suffices to reduce the evaluation problem for these expressions to CTI. The core of the reduction is the simple construction from [18,23] described in [5] (page 45), for the purpose of simulating ANDs and ORs using graph isomorphism questions. We adapt this construction as follows. Consider four trees trees G_1, G_2, H_1, and H_2 colored with two colors black and white (represented by black and white dots in the figures).

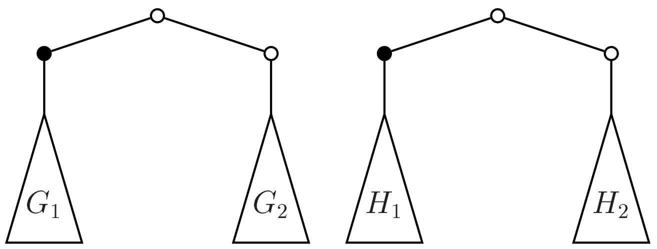

Fig. 1. Colored Trees for simulating AND

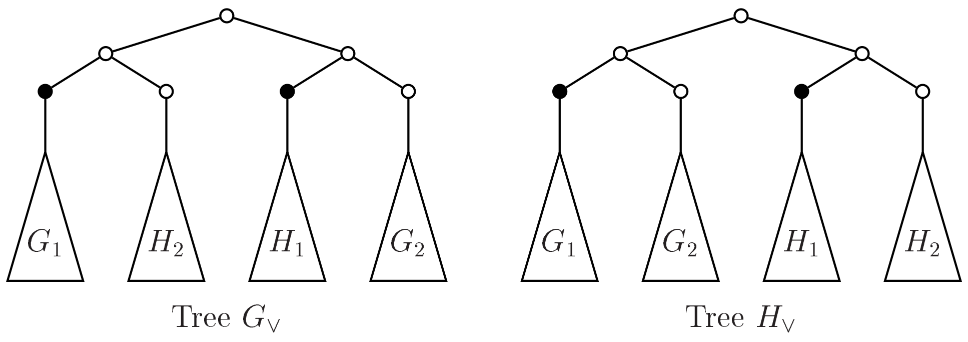

Fig. 2. Colored Trees for simulating OR

The colored trees in Figure 1 have the property that

$$G_\wedge \simeq H_\wedge \iff [(G_1 \simeq H_1) \wedge (G_2 \simeq H_2)]$$

and the colored trees in Figure 2 have the property that

$$G_\vee \simeq H_\vee \iff [(G_1 \simeq H_1) \vee (G_2 \simeq H_2)].$$

Observe that the OR-construction doubles the size of the initial trees for $G_1 \simeq G_2$ and $H_1 \simeq H_2$, that is, from 4 trees each having the same size, s, the OR-construction would produce 2 trees each having size $4s + 7$.

Now pick two single-node trees T_1 and T_2 and assign them different colors. Starting from the CTI instance (T_1, T_1) to represent a TRUE input and the CTI instance (T_1, T_2) to represent a FALSE input, it is trivial to construct, from a depth-d Boolean expression f with no negation gates, two colored trees G and H having $O(4^d)$ nodes (this is a rough upper bound) with the property that $G \simeq H$ iff f evaluates to TRUE.

To see that this yields an $\mathbf{AC}^0$ reduction, note it is possible to add dummy nodes in order to modify the constructs above in such a way that, if G_1, G_2, H_1 and G_2 are full binary (colored) trees, then so are $G_\wedge$, $H_\wedge$, $G_\vee$ and $H_\vee$, and moreover, the color occurrences in these respective constructs are the same. Because the Boolean expression is balanced, its depth is $O(\log n)$. And because the expression is built from alternating levels of ANDs and ORs, the structure of the resulting CTI instance is very regular (in a non technical sense). This makes it easy to translate the encoding of the source DLOGTIME-uniform formula into an encoding of the CTI instance having the property that the information relevant to a node can be deduced from the node encoding (just as properties of a subformula were trivially deduced from its encoding). Details of a similar construction can be found in the proof of [20, Lemma 6.2]. ■

Remark 1. The complete simulation used in the proof of Lemma 3.1 in fact requires only two distinct colors. A similar construction could be devised in the absence of colors as well.

Remark 2. It is easy to make an OR-construction for the tree automorphism problem. However no AND-construction for this problem is known. Because of this fact it is not known whether tree automorphism under the string representation is hard for $\mathbf{NC}^1$.

Lemma 3.2 *In the string representation,* CTI $\leq_m^{\mathbf{AC}^0}$ TI.

Proof. The obvious idea is to simulate the colors by attaching color-dependent gadgets at each node. Suppose that the trees in the CTI instance have n nodes. Then it suffices to attach at each c-colored node, $1 \leq c \leq n$, a new node which is root to a height-one subtree having $n + c$ leaves. In detail, at the string encoding level, the color binary number c occurring after the opening bracket which specifies the occurrence of the c-colored node is simply replaced with the encoding of the c-color gadget. To ensure $\mathbf{AC}^0$-computability, the color gadgets are modified to contain an identical number of nodes: it is easy to implement the idea with n non-isomorphic gadgets each having $2n + 1$ nodes. ■

Theorem 3.3 *In the string representation,* CTI *and* TI *are* $\mathbf{NC}^1$*-complete under* $\leq_m^{\mathbf{AC}^0}$.

Proof. CTI is $\mathbf{NC}^1$-hard (Lemma 3.1) and CTI $\leq_m^{\mathbf{AC}^0}$ TI (Lemma 3.2). Buss [8] shows in a delicate argument that TI $\in \mathbf{NC}^1$. (Buss in fact points out that his $\mathbf{NC}^1$ algorithm applies directly, as well, to the case of labeled trees.) Hence CTI is $\mathbf{NC}^1$-complete.

$\mathbf{NC}^1$-hardness of TI follows by the transitivity of the $\mathbf{AC}^0$-reducibility. ■

4 Tree Isomorphism for Trees Given by Pointers

Recall Lemma 3.2, which states that in the string representation, colored tree isomorphism reduces to tree isomorphism by the introduction of appropriate gadgets for the colors. These gadgets are clearly $\mathbf{AC}^0$-computable in the pointer representation, proving the following:

Lemma 4.1 *In the pointer representation,* CTI $\leq_m^{\mathbf{AC}^0}$ TI. ■

In the pointer representation not only the isomorphism, but also the automorphism problem is complete for L. Let us denote by TA the automorphism problem restricted to trees, and by CTA the colored version of the problem. A variation of the above Lemma shows that CTA and TA are $\mathbf{AC}^0$ equivalent problems.

Theorem 4.2 *In the pointer representation,* TI *and* TA *are* **L***-complete under* $\leq_m^{\mathbf{AC}^0}$.

Proof. The containment for TI follows from Lindell [7], who shows that a canonical form $c(T)$ of a (rooted) tree T can be computed in logspace. Hence, for two given trees, we determine isomorphism by computing and comparing their canonical forms symbol by symbol. (Alternatively, the string representation of the trees could be computed in **L**, and then Buss's $\mathbf{NC}^1$ algorithm could be used.) TA is $\mathbf{AC}^0$ reducible to TI. This result was shown in [18] for general graphs and can be adapted to trees using the constructions from the previous section. From this observation it follows that TA belongs to L. We prove hardness for **L** of TA and TI. Using the mentioned reduction from TA to TI it would suffice to show hardness for TA. We give however a direct argument showing the hardness for **L** of TI in a simple way.

We prove first hardness of TA by reducing the **L**-complete problem ORD [24] to Colored Tree Automorphism, and appealing to a variation of Lemma 4.1 for tree automorphism:

Order between Vertices (ORD)
Given: A digraph $G = (V, E)$ that is a line, and two nodes $v_i, v_j \in V$.
Problem: Decide whether $v_i < v_j$ in the total order induced on V.

Let a line graph $G = (V = \{v_1, \dots, v_n\}, E)$ and two designated nodes v_i and $v_j \in V$ be given. We assume without loss of generality that v_n is the out-degree zero node and that v_i, v_j and v_n are three different nodes. Let T' be the colored tree that results from G by coloring the node v_i with color 1, v_j with color 2, v_n with color 3, and the rest of the nodes with color 0. For $1 \leq k < l < n$, the colored tree $T_{k,l}$ is defined as $(V, \{(v_m, v_{m+1}) | 1 \leq m < n\})$, with the nodes v_k, v_l and v_n colored with colors 1,2 and 3 respectively, and the rest of the nodes colored by 0.

It is not hard to see that $v_i < v_j$ in the order induced on V by G if and only if $\exists k, l$ with $1 \leq k < l < n$ such that $T' \simeq T_{k,l}$. This is equivalent to saying that the graph $G = T' \cup \bigcup_{1 \leq k < l < n} T_{k,l}$ has a nontrivial automorphism. We can transform G to a tree by considering a new root node r and joining r with the v_1 nodes of all the $T_{k,l}$ subgraphs. This shows that ORD is $\mathbf{AC}^0$ reducible to CTA.

We reduce now ORD to CTI. Let again a line graph $G = (V = \{v_1, \dots, v_n\}, E)$ and two designated nodes v_i and $v_j \in V$ be given. W.l.o.g. consider v_{i+1} to be the successor of v_i in G. Let T' be the tree that results by making two copies v'_k, v''_k of each node v_k in G, and including the edges (u', v') (u'', v'') for each edge $(u,v) \in E$ with $u \neq v_i$ and the edges (v'_i, v'_{i+1}) (v'_i, v''_{i+1}). We also add a new root node r and the edges $(r, v'_1), (r, v''_1)$.

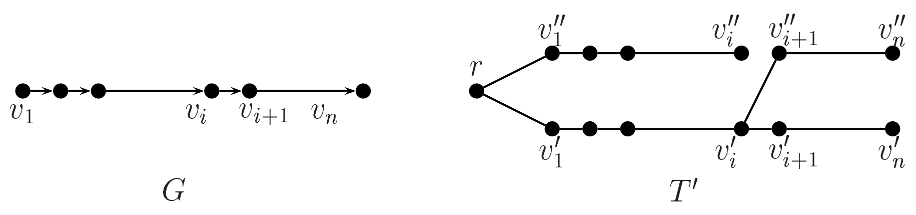

Fig. 3. The tree T'

Clearly $v_i < v_j$ in the order induced on V by G if and only if there is an automorphism in T' mapping v'_j to v''_j. This question can be reduced to CTI by making two copies T_1 and T_2 of T' and marking with a special color node v'_j in T_1 and node v''_j in T_2. It follows that $v_i < v_j$ in the order induced on V by G if and only if T_1 is isomorphic to T_2. ■

Remark. The completeness results have been stated in terms of rooted trees with the edges only implicitly directed. It is not hard to adapt the above proofs to show that TI and TA are also hard for **L** for the cases of unrooted trees or rooted trees with explicitly directed edges. The completeness of the problems follows by the fact that Lindell's algorithm [7] can be adapted to unrooted trees just by considering all possible roots.

5 Upper bound for 2-GA

We show in this section that 2-GA is reducible to the problem $\overline{\text{UCC}}$ of deciding whether a given graph has more than a single connected component.

The next lemma shows that we can restrict ourselves to graphs with at most two edges between any two colors. The proof of this result is straightforward.

Lemma 5.1 *Let $G = (V, E)$ be a graph with colored vertices, and C_i and C_j two color classes in G. Then $Aut(G) = Aut(G')$, where $G' = (V, E')$ is a copy of G but with the dual set of edges between vertices of C_i and C_j (for every pair (u,v) with $u \in C_i$ and $v \in C_j$, it holds $(u,v) \in E \Leftrightarrow (u,v) \notin E'$).*

Theorem 5.2 *The Graph Automorphism problem restricted to graphs with color classes of size at most2 is many-one $\mathbf{AC}^0$-reducible to $\overline{\text{UCC}}$.*

Proof. Let G be a graph with color classes of size at most 2. By the above lemma we can consider that for every pair of colors, there are at most two edges connecting the nodes of these colors. Also, w.l.o.g. we can consider that there is no edge between the nodes of the same color in G (these edges do not change the automorphism group). The possible connections that have to be considered between the nodes of two different colors are shown in Figure 4.

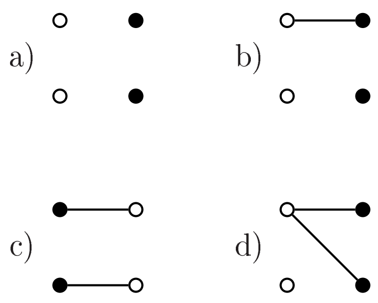

Fig. 4. Possible connections between two color classes

We reduce the question of whether G has a nontrivial automorphism to a reachability question in an undirected graph $f(G) = (V', E')$. V' contains one special node plus one node for each color:

$$V' = \{v_{id}\} \cup \{v^i \mid i \text{ is a color in } G\}.$$

The idea of the reduction is to place the edges in such a way that for every color i, node v_{id} is reachable from node v^i in $f(G)$ if and only if every automorphism in $Aut(G)$ fixes the nodes of color i. For every pair of colors i, j, the edges between two color classes induce edges in E' in the following way:

Type of connection between i and j	Edges in E'
a)	None
b)	$(v^i, v_{id}), (v^j, v_{id})$
c)	(v^i, v^j)
d)	(v^i, v_{id}).

Also, if there is only one node of color i, the edge (v^i, v_{id}) is included in E'.

Lemma 5.3 *For each color i, one can reach node v_{id} from v^i in $f(G)$ if and only if every automorphism in $Aut(G)$ fixes the nodes of color i.*

Proof. The proof from left to right is by induction on the number of edges in a path between v^i and v_{id}. If the path has length one, then the connections between the nodes of color i and some other color class in G must be like in cases b) or d). In any of these cases the nodes of color i must be fixed under any automorphism in G. If the path of minimal length has $k+1$ edges then there must be a color j and a path with k edges from v^j to v_{id} in $f(G)$. Also there must be an edge (v^i, v^j) in $f(G)$ which means that the connections between color classes i and j in G must be like in case c). By induction hypothesis every automorphism in $Aut(G)$ fixes the nodes of color j, and the connection c) forces the nodes of color i to be fixed also.

For the other direction, suppose that for some color i node v_{id} is not reachable from v^i in $f(G)$. We will show that in this case there is some non-trivial automorphism in G. W.l.o.g. let $\{1,\ldots,k\}$ be the set of colors j such that v^j is connected to v^i in $f(G)$. Since v_{id} is not reachable from v^i, in case there are edges between two classes with colors in $\{1,\ldots,k\}$ in G, these edges form a connection of type c), and the only possible edges between classes of colors l and m, with $l \notin \{1,\ldots,k\}$ and $m \in \{1,\ldots,k\}$ are connections of type d). We claim that the color fixing permutation φ interchanging the two nodes of the classes of colors $j \in \{1,\ldots,k\}$ and fixing the rest of the nodes is a non-trivial automorphism in G. Let u, v be two nodes in V. If neither u nor v have colors in the set $\{1,\ldots,k\}$ then φ acts like the identity on (u,v). If both u and v have colors l, m in the set $\{1,\ldots,k\}$ then either there is no connection or the connections between the color classes l and m must be of type c). It follows that $(u,v) \in E$ if and only if $(u',v') \in E$, where u' and v' are the other nodes with colors l and m respectively. By the definition of φ, $(u',v') = (\varphi(u), \varphi(v))$. Finally, if u has color $l \notin \{1,\ldots,k\}$ and v has color $m \in \{1,\ldots,k\}$ and there is a connection between nodes of color l and m then it must be of type d) (a connection of type c) would force color l to be in $\{1,\ldots,k\}$). By definition $(\varphi(u), \varphi(v)) = (u, v')$. As in the previous two cases we have $(u,v) \in E$ if and only if $(\varphi(u), \varphi(v)) \in E$. ■

Observe that Lemma 5.3 implies that the graph G has a nontrivial automorphism if and only if $f(G)$ has more than one connected component. ■

6 Upper bounds for 3-GI and 3-GA

We deal in this section with graphs having up to 3 different nodes of each color. We denote by B_3 the set of 6 bijections defined over a domain set of 3 elements. An isomorphism between graphs with color classes of size 3 can be decomposed in a product of bijections from B_3.

Theorem 6.1 *The Graph Isomorphism problem restricted to graphs with color classes of size at most 3 is in the class* **SL**.

Proof. Let G and H be the input graphs. By Lemma 5.1 we can consider that for any pair of colors i, j there are at most 4 edges in each one of the input graphs having as endpoints nodes of colors i and j. If there are more than 4 edges we consider the dual connections. The three nodes of a color i in G are labeled i_1, i_2 and i_3, and in H by i'_1, i'_2 and i'_3.

We will reduce graphs G and H to a single undirected graph $f(G,H)$ and translate the isomorphism question to a reachability question in $f(G,H)$. For each color i we will consider 6 nodes in $f(G)$ corresponding to the possible bijections in B_3. Additionally there is one extra node w that will be used to indicate that some isomorphisms between subgraphs of G and H are not possible. The set of nodes in $f(G,H)$ is:

$$\{v^i_\varphi \mid \varphi \in B_3 \text{ and } i \text{ is a color in } G\} \cup \{w\}.$$

The set of edges in $f(G,H)$ is constructed according to the following 2 rules. Observe that both rules can be applied in logarithmic space since the color classes have constant size 3 and there can only be at most 6 potential isomorphisms between the nodes of a given color.

Let C_i (resp. C'_i) denote the set of nodes of color i in G (resp. in H). For each color i the edges with both endpoints in C_i or C'_i might imply that some of the bijections from C_i to C'_i cannot be extended to an isomorphism between G and H. The restrictions in the set of possible bijections can also be induced by the connections with a different color class. In these cases we include some edges in the graph $f(G,H)$ as in rule 1, indicating that a bijection between the nodes of a color cannot be extended to an isomorphism:

Rule 1: For every pair of colors i, j and every bijection $\varphi \in B_3$, we include in $f(G,H)$ the edge (v^i_φ,w) if the edges between C_i and C_j in G and the edges between C'_i and C'_j in H imply that no isomorphism in $\mathrm{Iso}(G,H)$ can map the nodes of C_i into C'_i like φ, that is, if for every $\psi \in B_3$, $\varphi \times \psi \notin \mathrm{Iso}(C_i \cup C_j, C'_i \cup C'_j)$, where $(C_i \cup C_j)$ is the graph made by the set of nodes C_i and C_j and the edges between them. For the trivial case in which there is only one color i in G, the edge (v^i_φ, w) is included in $f(G,H)$ when $\varphi \notin \mathrm{Iso}(C_i, C'_i)$.

We also include in $f(G,H)$ some edges between nodes corresponding to different color classes indicating that a partial isomorphism between the nodes of some color forces a partial isomorphism between the nodes of another color.

Rule 2: For every pair of colors i, j, and $\varphi \in B_3$. If for a pair of nodes $a, b \in \{i_1, i_2, i_3\}$ and a pair of nodes $a', b' \in \{i'_1, i'_2, i'_3\}$ and two bijections $\eta, \pi \in B_3$ the edges between the sets of nodes $\{a, b\}$ and $\{\eta(a), \eta(b)\}$ are exactly the two edges $\{(a, \eta(a)), (b, \eta(b))\}$ and the edges between the sets of nodes $\{a', b'\}$ and $\{\pi(a'), \pi(b')\}$ are exactly $\{(a', \pi(a')), (b', \pi(b'))\}$ and $\varphi \times \psi \in$

$\text{Iso}(C_i \cup C_j, C_i' \cup C_j')$ (for $\psi = \pi\varphi\eta^{-1}$) then we include in E' the edge (v_φ^i, v_ψ^j) (see Figure 5).

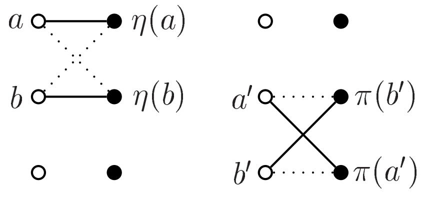

Fig. 5. A situation in which rule 2 is applied. (The dotted lines indicate that these edges do not exist.)

We show with the following lemmas that a reachability question in $f(G, H)$ can be used to decide whether G and H are isomorphic.

Lemma 6.2 *For a pair of colors i, j and $\varphi, \psi \in B_3$, if $(v_\varphi^i, w) \notin f(G, H)$ and $(v_\psi^j, w) \notin f(G, H)$ and $(v_\varphi^i, v_\pi^j) \notin f(G, H)$ for any $\pi \in B_3$, then $\varphi \times \psi$ is an isomorphism between the subgraphs $C_i \cup C_j$ and $C_i' \cup C_j'$.*

Proof. Let $G = (V, E)$ and $H = (W, E)$ be the input graphs and let us denote by a, b and c the nodes of color i in G, by d, e and f the color j nodes. We denote by the same symbols with ' the nodes of these colors in H. W.l.o.g we can suppose that for $l \in \{a, b, c\}$ $\varphi(l) = l'$ and for $m \in \{d, e, f\}$ $\psi(m) = m'$. Let us suppose by contradiction that $\varphi \times \psi \notin \text{Iso}(C_i \cup C_j, C_i' \cup C_j')$. It follows that for a pair of nodes a, d,

$$(a, d) \in E \Leftrightarrow (\varphi(a), \psi(d)) \notin F.$$

We consider that $(a, d) \in E$, the other case is analogous. Since v_φ^i and v_ψ^j are not connected to w, it follows by rule 1 that there are two bijections η and $\pi \in B_3$ such that $\varphi \times \eta$ and $\pi \times \psi \in \text{Iso}(C_i \cup C_j, C_i' \cup C_j')$. Because of these facts we have that $\eta(d) \neq d'$ and $\pi(a) \neq (a')$. Again w.l.o.g. we can suppose $\eta(d) = e'$ and $\pi(a) = b'$. It follows $(\varphi(a), \eta(d)) = (a', e') \in F$ and $(\pi(a), \psi(d)) = (b', d') \in F$. The rest of the proof consists in considering the different possibilities for the bijections η and π showing that in each case we reach a contradiction.

Case 1: Suppose $\eta(d) = e', \eta(e) = d'$ and $\eta(f) = f'$. $(a', d') \notin F$ implies $(\varphi^{-1}(a'), \eta^{-1}(d')) = (a, e) \notin E$ and therefore $(\pi(a), \psi(e)) = (b', e') \notin F$ and also $(\varphi^{-1}(b'), \eta^{-1}(e') = (b, d) \notin E$. If $\pi(b) = a'$ then from $(a', e') \in F$ follows $(\pi^-1(a'), \psi^{-1}(e')) = (b, e) \in E$. But then we have that the edges between the nodes a, b, d, e in G and between their counterparts in H are exactly as is rule 2, and there should be an edge in $f(G, H)$ from v_φ^i to some node of color j

contradicting the hypothesis. The other possible situation in when $\pi(b) = c'$ and $\pi(c) = a'$ but then from $(b, d) \notin F$ follows $(\varphi^{-1}\pi(b), \eta^{-1}\psi(d)) = (c, e) \notin E$, and from $(a', e') \in F$ follows $(\pi^{-1}(a'), \psi^{-1}(e')) = (c, e) \in E$ which is a contradiction.

Case 2: Suppose $\eta(d) = e', \eta(e) = f'$ and $\eta(f) = d'$. By the same arguments as in Case 1, we have $(b, f) \in E$, $(a, f) \notin E$ $(b', f') \notin F$ and $(b', e') \notin E$. If $\pi(b) = a'$ and $\pi(c) = c'$ then from $(b, e) \notin E$ follows $(\pi(b), \psi(e)) = (a', e') \notin F$ which is a contradiction. Finally, if $\pi(b) = c'$ and $\pi(c) = a'$ then it follows $(a, d), (b, f), (c, e) \in E$, $(a, f), (b, e), (c, d) \notin E$, $(a', e'), (b', d'), (c', f') \in F$ and $(a', d'), (b', f'), (c', e') \notin F$. Consider the pair of nodes a and e. If $(a, e) \in E$ then $(a', f'), (b', e') \in F$ contradicting the fact that there are at most 4 edges in H connecting the i and j nodes. On the other hand, if $(a, e) \notin E$ then the set of edges between the i and j color nodes are exactly $(a, d), (b, f), (c, e)$ in E and $(a', e'), (b', d'), (c', f')$ in F. By rule 2 there should be some edge in $f(G, H)$ from v^i_φ to some node of color j, contradicting the hypothesis. ∎

Lemma 6.3 *For each pair of colors i, j, and $\varphi, \psi \in B_3$, if there is a path from v^i_φ in $f(G, H)$ to v^j_ψ not having w as an intermediate node then every isomorphism in Iso(G, H) that maps the nodes of color i like φ, is forced to map the nodes of color j like ψ.*

Proof. We use induction on the length of a minimal path from v^i_φ to v^j_ψ in $f(G, H)$. If this path has length 1 then the colors i and j are different and the edge (v^i_φ, v^j_ψ) in $f(G, H)$ has been placed by rule 2. This implies that for some $a, b \in \{1, 2, 3\}$ and some $c, d \in \{1, 2, 3\}$ the edges $(i_a, j_{\eta(a)})$, $(i_b, j_{\eta(b)})$ (for some $\eta \in B_3$) are the only edges between the sets of nodes $\{i_a,\ i_b\}$ and $\{j_{\eta(a)}, j_{\eta(b)}\}$ in G and the edges $(i_a, j_{\pi(a)})$, $(i_b, j_{\pi(b)})$ (for some $\pi \in B_3$) are the only edges between the sets of nodes $\{i'_c,\ i'_d\}$ and $\{j_{\pi(c)}, j_{\pi(d)}\}$ in H. Moreover, because of rule 2, $\psi = \pi\varphi\eta^{-1}$. If an isomorphism in Iso(G, H) maps the i nodes to the i' nodes like φ, then for $l \in \{\eta(a), \eta(b)\}$ j_l must be mapped to a node j' in H connected to $i'_{\varphi\eta^{-1}(l)}$, and this node is $j_{\pi\varphi\eta^{-1}(l)} = j_{\psi(l)}$. It follows also for $c \notin \{a, b\}$, j_c is mapped to $j_{\pi\varphi\eta^{-1}(c)}$.

For the induction step, if the number of edges in the path from v^i_φ and v^j_ψ in $f(G, H)$ is $k + 1$, let m be the first color after i in the path. There has to be some bijection $\phi \in B_3$ and an edge $(v^i_\varphi$,$v^m_\phi)$ between the i nodes and the j nodes in $f(G, H)$ and this edge must be introduced by rule 2. By induction hypothesis, every isomorphism in Iso(G, H) mapping the i nodes like φ must map the m nodes like ϕ, and every isomorphism in Iso(G, H) mapping the m nodes like ϕ must map the j nodes like ψ. Both conditions together imply the result. ∎

Observe that Lemma 6.3 implies that for any color i, and $\varphi, \psi \in B_3$, if $\varphi \neq \psi$, and v^i_φ is reachable from v^i_ψ in $f(G, H)$ then there is no isomorphism in

$\mathrm{Iso}(G,H)$ mapping the nodes of color i like φ. Another consequence of the lemma (together with the definition of rule 1), is that for any color i and $\varphi \in B_3$, if node w can be reached from v^i_φ in $f(G,H)$ then there is no isomorphism in $\mathrm{Iso}(G,H)$ mapping the nodes of color i like φ.

Lemma 6.4 *Suppose that there are k different colors in G and H. If there is a set of nodes in $f(G,H)$, $v^1_{\varphi_1},\ldots,v^k_{\varphi_k}$ one of each color, and such that no other node in $f(G,H)$ is reachable from this set then $\varphi_1 \times \ldots \times \varphi_k$ is an isomorphism between G and H.*

Proof. The proof is by induction on the number of color classes k in G and H. If there is only one color the result is trivial. For the case of two colors i and j, consider that from the set of nodes v^i_φ and v^j_ψ one cannot reach any other node in $f(G,H)$. This implies that the only possible edge in $f(G,H)$ with an endpoint in this set is the one connecting both nodes. If this edge does not exist the result follows by Lemma 6.2. On the other hand, if (v^i_φ, v^j_ψ) is an edge in $f(G,H)$ then the edge was placed by rule 2 and $\varphi \times \psi \in \mathrm{Iso}(G,H)$.

For the induction step, consider that there are k colors in G and H and there is a set of nodes one of each color $v^1_{\varphi_1},\ldots,v^k_{\varphi_k}$ in $f(G,H)$. Consider the graphs G' and H' obtained by deleting the nodes of color k in G and H and all the edges having an endpoint of this color. Since eliminating one color can only reduce the set of local restrictions for isomorphisms, there is no new edge in $f(G',H')$ that was not already present in $f(G,H)$ and therefore from $v^1_{\varphi_1},\ldots,v^k_{\varphi_{k-1}}$ no other node is reachable in $f(G',H')$. By induction hypothesis $\varphi_1 \times \ldots \times \varphi_{k-1} \in \mathrm{Iso}(G',H')$. We claim that this isomorphism between G' and H' can be extended to an isomorphism in $\mathrm{Iso}(G,H)$ by mapping the nodes in C_k to C'_k as in φ_k.

To see that this is an isomorphism we will show that for every $j < k$ it holds $\varphi_k \times \varphi_j \in \mathrm{Iso}(C_k \cup C_j, C'_k \cup C'_j))$. If the edge $(v^k_{\varphi_k}, v^j_{\varphi_j})$ belongs to $f(G,H)$ then the edge was placed by rule 2 and $\varphi_k \times \varphi_j \in \mathrm{Iso}(C_k \cup C_j, C'_k \cup C'_j))$. On the other hand if this edge does not exist, there is no other edge in $f(G,H)$ between k and j nodes having $v^k_{\varphi_k}$ or $v^j_{\varphi_j}$ as endpoint (from the set $v^1_{\varphi_1},\ldots,v^k_{\varphi_k}$ no other node can be reached). The result follows then by Lemma 6.2. ∎

It follows from Lemmas 6.3 and 6.4 that there is an isomorphism from G to H if and only if there is a set of nodes in $f(G,H)$, one of each color and such that from this set no other node in $f(G,H)$ can be reached. In order to transform this into a question in **SL** we need the following lemma:

Lemma 6.5 *Let A and B be two connected components in $f(G,H)$ satisfying*

i) The nodes in A have different colors and the nodes in B have different colors

ii) $w \notin A \cup B$ and
iii) the intersection of colors in A and B is not empty

then the set of colors present in A is the same as the set of colors present in B.

Proof. We show that for any pair of colors i and j, and for two bijections φ and $\xi \in B_3$, if the node w in $f(G,H)$ can neither be reached from v^i_φ nor from v^i_ξ and if v^i_φ has a neighbor of color j in $f(G,H)$, then so does v^i_ξ. The result follows from this fact.

Suppose that (v^i_φ, v^j_ψ) is an edge in $f(G,H)$. This edge was set by rule 2 and therefore there is a pair of nodes $a, b \in \{i_1, i_2, i_3\}$ and a pair of nodes $a', b' \in \{i'_1, i'_2, i'_3\}$ and two bijections $\eta, \pi \in B_3$ satisfying the conditions of rule 2 and such that $\psi = \pi\varphi\eta^{-1}$. Since v^i_ξ is not connected to w, there must be a bijection $\gamma \in B_3$ satisfying $\xi \times \gamma \in \text{Iso}(C_i \cup C_j, C'_i \cup C'_j)$. η and π describe the connections between i and j nodes in G and H and therefore we have $\gamma = \pi\xi\eta^{-1}$. We are again in the conditions of rule 2, and the edge (v^i_ξ, v^j_γ) belongs to $f(G,H)$ ■

We know that there is an isomorphism from G to H if and only if there is a set of nodes in $f(G,H)$, one of each color and such that from this set no other node in $f(G,H)$ can be reached. By Lemma 6.5 in order to test this it suffices to check for each color i that there is one node v^i_φ in $f(G,H)$ from which neither w nor two nodes of the same color can be reached (this question can be solved within the class **SL**). In order to see this observe that if G and H are isomorphic, such a set must exist. By Lemma 6.5 if such a set exists then there is a set of color disjoint connected components in $f(G,H)$ containing all colors. ■

We show now that the graph automorphism problem for colored graphs with color classes of size 3 also lies in the class **SL**. Although a direct proof similar to the one in Theorem 6.1 is possible, it is easier to give a reduction from 3-GA to UGAP based on the fact that 3-GI $\in$ **SL**.

Theorem 6.6 *The Graph Automorphism problem restricted to graphs with color classes of size at most 3 is in the class* **SL**.

Proof. We will show that 3-GA is logarithmic space many-one reducible to UGAP. This implies that 3-GA lies in **SL**. Let $G = (V, E)$ be a graph with its nodes partitioned into color classes of size at most three. We denote by $G_{[i]}$ a copy of G but with node i marked with a new special color. There is a non-trivial automorphism in G if and only if for a pair of distinct nodes of the same color i and j in V, there is an automorphism mapping i to j, if and only if for such a pair of nodes $G_{[i]}$ is isomorphic to $G_{[j]}$. Since the color classes in $G_{[i]}$ have size 3 at most, this means that 3-GA is reducible to a set of disjunctive queries to 3-GI. By Theorem 6.1, 3-GI lies in **SL** and can be reduced to UGAP. The list of queries to 3-GI can then be reduced to a list of reachability queries in undirected graphs. The disjunctive list of queries to UGAP can be reduced to a single one by connecting the graphs in parallel. This provides a many-one reduction from 3-GA to UGAP. ■

7 Lower bounds for 2-GI and 2-GA

We prove now the hardness results for 2-GI and 2-GA.

Theorem 7.1 2-GI *is hard for* **SL** *under* $\leq_m^{\mathbf{AC}^0}$.

Proof. We show that the graph accessibility problem for undirected graphs, UGAP, is reducible to the complement of 2-GI. The result follows since UGAP is $\mathbf{AC}^0$ complete for **SL**, and this class is closed under complementation [25].

Let $G = (V, E)$ be an undirected graph with two designated nodes $s, t \in V$. Consider the new graph $G' = G_1 \cup G_2$ where G_1 and G_2 are two copies of G, and for a node $v \in V$ let us call v_1 and v_2 the copies of v in G_1 and G_2 respectively. Furthermore, we color each pair of nodes v_1, v_2 with color i_v, and color t_1 with a special color 1 and t_2 with another color 2. We claim that there is no path from s to t in G if and only if there is automorphism φ in G' mapping s_1 to s_2. Clearly if there is no path between s and t in G, these two nodes belong to different connected components. The desired automorphism can be obtained by mapping the nodes of the connected component of s_1 in G_1 to the corresponding nodes in G_2 and mapping the rest of the nodes in G' (and in particular t_1) to themselves.

Conversely, if there is a path between s and t in G, the mentioned automorphism φ does not exist since the nodes s_2 $(\varphi(s_1))$ and t_1 $(\varphi(t_1))$ should be in the same connected component, but there are no edges between G_1 and G_2 in G'.

The question of whether there is an automorphism in G' with the mentioned properties, can in turn be reduced to 2-GI. Let H_1 be a copy of G' with node s_1 having a special color 3, and H_2 be another copy with s_2 having color 3. There is an automorphism mapping s_1 to s_2 in G' if and only if H_1 and H_2 are isomorphic. The size of the color classes in each of the graphs H_1 and H_2 is at most 2. ■

Now the hardness result for 2-GA follows easily.

Corollary 7.2 $\overline{\text{UCC}} \leq_m^{\mathbf{AC}^0}$ 2-GA.

Proof. The reduction for this result is the same as in the proof of the previous theorem. Observe that in G' the color classes are of size at most 2 and there is a nontrivial automorphism in G' if and only if there is more than a single connected component in G. ■

8 Concluding remarks

Together with the upper bounds given by Buss and Lindell our results imply:

- TI in the string representation is $\mathbf{NC}^1$-complete under $\mathbf{AC}^0$ reducibility.
- TI and TA in the pointer representation are $\mathbf{NC}^1$-complete under $\mathbf{AC}^0$ reducibility.
- 2-GA is equivalent to $\overline{UCC}$ under $\mathbf{AC}^0$ reducibility.
- 2-GI and 3-GI are complete for **SL** under $\mathbf{AC}^0$ reducibility.
- 3-GA belongs to **SL**.

The level of sophistication of Buss's $\mathbf{NC}^1$ algorithm for TI [8] is comparable to that of his simplified $\mathbf{NC}^1$ algorithm for the Boolean expression value problem FVP [13]. Are these two upper bounds independent? In other words, is there a reduction from TI to FVP or vice versa which is simpler than either of Buss's two upper bounds?

It is interesting to consider FVP $\leq_m^{\mathbf{AC}^0}$ TI. Proving that FVP $\leq_m^{\mathbf{AC}^0}$ TI has required three ingredients: (1) the $\mathbf{NC}^1$ upper bound for FVP, (2) the characterization of $\mathbf{NC}^1$ in terms of balanced Boolean expressions, and (3) our simple Lemma 3.1. Lemma 3.1 directly constructs trees from Boolean formulas, but the ensuing direct reduction is from *Balanced*-FVP to TI. How can Lemma 3.1 be strengthened?

The bottleneck to a strengthening of Lemma 3.1 is the handling of a Boolean OR. Lemma 3.1 can only handle balanced Boolean expressions because the trees $G_\vee$ and $H_\vee$ depicted in its proof each require a copy of G_1, G_2, H_1, and H_2. Hence an open question is whether Lemma 3.1 can be proved using simpler constructs $G_\vee$ and $H_\vee$, still simulating the Boolean OR, but only adding a small number of additional nodes. If so, the $\mathbf{NC}^1$ upper bound for FVP is redundant, i.e., the $\mathbf{NC}^1$ upper bound for FVP follows from the $\mathbf{NC}^1$ upper bound for TI.

A natural way to continue this research is to study the situation for other bounds $b \geq 4$ for the size of the color classes trying to obtain completeness results for other complexity classes. ¿From the results in [10] it can easily be derived that for $b \geq 2$, b^2-GI is hard for the modular class $\mathbf{Mod}_b\mathbf{L}$, and $2b^2$-GA is also hard for $\mathbf{Mod}_b\mathbf{L}$. It follows from this that 4-GI is hard for $\oplus\mathbf{L}$ and therefore a proof of the fact that 4-GI belongs to **SL** would imply that $\oplus\mathbf{L}$ is included in **SL** which is something we do not expect. Obtaining better upper bounds than the ones given in [9] for b-GI and b-GA for special cases of k ($k \geq 4$) is an interesting open problem.

We observe that the blow-up in the complexity of the problem when going from color classes of size 3 to size 4 also happens in the related area of graph identification using first-order formulas with counting. Immerman and Lander show in [26] that 3 variables suffice to identify all colored graphs of color size 3, while $\Omega(n)$ variables are needed to identify all graphs of color size 4 using first order formulas with counting, as proved in [27].

Acknowledgments: We would like to thank Eric Allender, V. Arvind, Pascal Tesson and Denis Thérien for interesting discussions in the topics of the paper. We would also like to thank Klaus-Jörn Lange for several invitations to Tübingen, where part of this research was done.

References

[1] B. Jenner, P. McKenzie, J. Torán, A note on the hardness of tree isomorphism, in: Proc. 13th Annual IEEE Conference on Computational Complexity, IEEE Computer Society Press, 1998, pp. 101–106.

[2] J. Köbler, J. Torán, The complexity of graph isomorphism for colored graphs with color classes of size 2 and 3, in: Proc. 19th Symposium on Theoretical Aspects of Computer Science, Vol. 2285 of Lecture Notes in Computer Science, Springer-Verlag, Berlin Heidelberg New York, 2002, pp. 121–132.

[3] L. Babai, Moderately exponential bounds for graph isomorphism, in: Proc. International Symposium on Fundamentals of Computing Theory 81, Vol. 117 of Lecture Notes in Computer Science, Springer-Verlag, Berlin Heidelberg New York, 1981, pp. 34–50.

[4] E. Luks, Isomorphism of bounded valence can be tested in polynomial time, Journal of Computer and System Sciences 25 (1982) 42–65.

[5] J. Köbler, U. Schöning, J. Torán, The Graph Isomorphism Problem: Its Structural Complexity, Birkhäuser, Boston, 1993.

[6] J. E. Hopcroft, R. E. Tarjan, A V^2 algorithm for determining isomorphism of planar graphs, Information Processing Letters 1 (1971) 32–34.

[7] S. Lindell, A logspace algorithm for tree canonization, in: Proc. 24th ACM Symposium on Theory of Computing, ACM Press, 1992, pp. 400–404.

[8] S. Buss, Alogtime algorithms for tree isomorphism, comparison, and canonization, in: Computational Logic and Proof Theory, 5th Kurt Gödel Colloquium'97, Vol. 1289 of Lecture Notes in Computer Science, Springer-Verlag, Berlin Heidelberg New York, 1997, pp. 18–33.

[9] E. Luks, Parallel algorithms for permutation groups and graph isomorphism, in: Proc. 27th IEEE Symposium on the Foundations of Computer Science, IEEE Computer Society Press, 1986, pp. 292–302.

[10] J. Torán, On the hardness of graph isomorphism, in: Proc. 41st IEEE Symposium on the Foundations of Computer Science, IEEE Computer Society Press, 2000, pp. 180–186.

[11] A. V. Aho, J. E. Hopcroft, J. D. Ullman, The design and analysis of computer algorithms, Addison-Wesley, 1974.

[12] G. Miller, J. Reif, Parallel tree contraction part 2: Further applications, SIAM Journal on Computing 20 (1991) 1128–1147.

[13] S. R. Buss, The boolean formula value problem is in ALOGTIME, in: Proc. 19th ACM Symposium on Theory of Computing, 1987, pp. 123–131.

[14] D. A. Barrington, Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1, Journal of Computer and System Sciences 38 (1989) 150–164.

[15] L. Babai, Monte Carlo algorithms for graph isomorphism testing, Technical Report 79-10, Dép. Math. et Stat., Univ. de Montréal (1979).

[16] M. Furst, J. Hopcroft, E. Luks, Polynomial time algorithms for permutation groups, in: Proc. 21st IEEE Symposium on the Foundations of Computer Science, IEEE Computer Society Press, 1980, pp. 36–41.

[17] H. Lewis, C. Papadimitriou, Symmetric space-bounded computation, Theoretical Computer Science 19 (1982) 161–187.

[18] A. Lozano, J. Torán, On the nonuniform complexity of the graph isomorphism problem, in: K. Ambos-Spies, S. Homer, U. Schöning (Eds.), Complexity Theory, Current Research, Cambridge University Press, 1993, pp. 245–273, also in *Proceedings of the 7th Structure in Complexity Theory Conference*, pp. 118–129, June 1992.

[19] C. Álvarez, R. Greenlaw, A compendium of problems complete for symmetric logarithmic space, Journal of Computational Complexity 9 (2000) 73–95.

[20] D. Barrington, N. Immerman, H. Straubing, On uniformity within NC^1, Journal of Computer and System Sciences 41 (1990) 274–306.

[21] S. A. Cook, P. McKenzie, Problems complete for deterministic logarithmic space, Journal of Algorithms 8 (1987) 385–394.

[22] M. Beaudry, P. McKenzie, Circuits, matrices and nonassociative computation, Journal of Computer and System Sciences 50 (3) (1995) 441–455.

[23] R. Chang, J. Kadin, On computing boolean connectives of characteristic functions, Mathematical Systems Theory 28 (1995) 173–198.

[24] K. Etessami, Counting quantifiers, successor relations, and logarithmic space, in: Proc. 10th Structure in Complexity Theory Conference, IEEE Computer Society Press, 1995, pp. 2–11.

[25] N. Nisan, A. Ta-Shma, Symmetric logspace is closed under complement, in: Proc. 27th ACM Symposium on Theory of Computing, ACM Press, 1995, pp. 140–146, appeared also in *Chicago Journal of Theoretical Computer Science*, article 1, 1995.

[26] N. Immerman, E. Lander, Describing graphs: a first order approach to graph canonization, in: A. L. Selman (Ed.), Complexity Theory Retrospective, Springer-Verlag, Berlin Heidelberg New York, 1990, pp. 59–81.

[27] J. Cai, M. Fürer, N. Immerman, An optimal lower bound for the number of variables for graph identification, Combinatorica 12 (1992) 389–410.