TheoretiCS

On the Constant-Depth Circuit Complexity of Generating Quasigroups

Received Oct 8, 2024
Accepted Jun 9, 2025
Published Aug 22, 2025

Key words and phrases
Group Isomorphism, Minimum Generating Set, Membership, Circuit Complexity, quasiAC^0

Nathaniel A. Collins [a]
Joshua A. Grochow [b,c]
Michael Levet [d]
Armin Weiß [e]

a Department of Mathematics, Colorado State University, Louis R. Weber Building Room 233, 841 Oval Drive, Fort Collins, CO 80521, USA

b Department of Computer Science, University of Colorado Boulder, 1111 Engineering Dr, ECOT 717, 430 UCB, Boulder, CO 80309, USA

c Department of Mathematics, University of Colorado Boulder, Campus Box 395, Boulder, CO 80309 USA

d Department of Computer Science, College of Charleston, 66 George Street, Charleston, SC 29424, USA

e Universität Stuttgart, Institute for Formal Methods of Computer Science, Universitätsstraße 38, 70569 Stuttgart, Germany

ABSTRACT. We investigate the constant-depth circuit complexity of the Isomorphism Problem, Minimum Generating Set Problem (MGS), and Sub(quasi)group Membership Problem (Membership) for groups and quasigroups (=Latin squares), given as input in terms of their multiplication (Cayley) tables. Despite decades of research on these problems, lower bounds for these problems even against depth-2 AC circuits remain unknown. Perhaps surprisingly, Chattopadhyay, Torán, and Wagner (FSTTCS 2010; *ACM Trans. Comput. Theory*, 2013) showed that Quasigroup Isomorphism could be solved by AC circuits of depth $O(\log\log n)$ using $O(\log^2 n)$ nondeterministic bits, a class we denote $\exists^{\log^2 n}$FOLL. We narrow this gap by improving the upper bound for many of these problems to quasiAC^0, thus decreasing the depth to constant.

A preliminary version of this work appeared in the proceedings of ISSAC 2024 [**19**]. JAG and ML were partially supported by JAG's NSF CAREER award CISE-204775 during this work. AW was partially supported by the German Research Foundation (DFG) grant WE 6835/1-2.

Cite as Nathaniel A. Collins, Joshua A. Grochow, Michael Levet, Armin Weiß. On the Constant-Depth Circuit Complexity of Generating Quasigroups. TheoretiCS, Volume 4 (2025), Article 19, 1-39.

https://theoretics.episciences.org
DOI 10.46298/theoretics.25.19

In particular, we show that MEMBERSHIP can be solved in $\mathsf{NTIME}(\mathrm{polylog}(n))$ and use this to prove the following:

— MGS for quasigroups belongs to $\exists^{\log^2 n}\forall^{\log n}\mathsf{NTIME}(\mathrm{polylog}(n)) \subseteq \mathsf{quasiAC}^0$. Papadimitriou and Yannakakis (*J. Comput. Syst. Sci.*, 1996) conjectured that this problem was $\exists^{\log^2 n}$P-complete; our results refute a version of that conjecture for completeness under $\mathsf{quasiAC}^0$ reductions unconditionally, and under polylog-space reductions assuming EXP ≠ PSPACE. It furthermore implies that this problem is not hard for any class containing PARITY. The analogous results concerning PARITY were known for QUASIGROUP ISOMORPHISM (Chattopadhyay, Torán, & Wagner, *ibid.*) and SUBGROUP MEMBERSHIP (Fleischer, *Theory Comput.* 2022), though not for MGS.

— MGS for groups belongs to $\mathsf{AC}^1(\mathsf{L})$. Our $\mathsf{AC}^1(\mathsf{L})$ bound improves on the previous, very recent, upper bound of P (Lucchini & Thakkar, *J. Algebra*, 2024). Our $\mathsf{quasiAC}^0$ upper bound is incomparable to P, but has similar consequences to the above result for quasigroups.

— QUASIGROUP ISOMORPHISM belongs to $\exists^{\log^2 n}\mathsf{AC}^0(\mathsf{TIMESPACE}(\mathrm{polylog}(n), \log(n))) \subseteq \mathsf{quasiAC}^0$. As a consequence of this result and previously known AC^0 reductions, this implies the same upper bound for the ISOMORPHISM PROBLEMS for: Steiner triple systems, pseudo-STS graphs, LATIN SQUARE ISOTOPY, Latin square graphs, and Steiner $(t, t+1)$-designs. This improves upon the previous upper bound for these problems, which was $\exists^{\log^2 n}\mathsf{L} \cap \exists^{\log^2 n}\mathsf{FOLL} \subseteq \mathsf{quasiFOLL}$ (Chattopadhyay, Torán, & Wagner, *ibid.*; Levet, *Australas. J. Combin.* 2023).

— As a strong contrast, we show that MGS for arbitrary magmas is NP-complete.

Our results suggest that understanding the constant-depth circuit complexity may be key to resolving the complexity of problems concerning (quasi)groups in the multiplication table model.

1. Introduction

The GROUP ISOMORPHISM (GPI) problem is a central problem in computational complexity and computer algebra. When the groups are given as input by their multiplication (a.k.a. Cayley) tables, the problem reduces to GRAPH ISOMORPHISM (GI), and because the best-known runtimes for the two are quite close ($n^{O(\log n)}$ [43][1] vs. $n^{O(\log^2 n)}$ [5][2]), the former stands as a key bottleneck towards further improvements in the latter.

Despite this, GPI seems quite a bit easier than GI. For example, Tarjan's $n^{\log n + O(1)}$ algorithm for groups [43] can now be given as an exercise to undergraduates: every group is generated by at most $\lfloor \log_2 |G| \rfloor$ elements, so the algorithm is to try all possible $\binom{n}{\log n} \leq n^{\log n}$ generating sets, and for each, check in $n^{O(1)}$ time whether the map of generating sets extends to an isomorphism.

1 Miller [43] credits Tarjan for $n^{\log n + O(1)}$.

2 Babai [5] proved quasi-polynomial time, and the exponent of the exponent was analyzed and improved by Helfgott [29].

In contrast, the quasi-polynomial time algorithm for graphs was a tour de force that built on decades of research into algorithms and the structure of permutation groups. Nonetheless, it remains unknown whether the problem for groups is actually easier than that for graphs under polynomial-time reductions, or even whether both problems are in P!

Using a finer notion of reduction, Chattopadhyay, Torán, and Wagner [17] proved that there was no AC^0 reduction from GI to GpI. This gave the first unconditional evidence that there is *some formal sense* (namely, the AC^0 sense) in which GpI really is easier than GI. The key to their result was that the generator-enumeration technique described above can be implemented by non-deterministically guessing $\log^2 n$ bits (describing the $\log n$ generators, each of $\log n$ bits), and then verifying an isomorphism by a non-deterministic circuit of depth only $O(\log\log n)$, a class we denote $\exists^{\log^2 n}\mathsf{FOLL}$. Observe that $\exists^{\log^2 n}\mathsf{FOLL} \subseteq \mathsf{quasiFOLL}$ (by trying all $2^{O(\log^2 n)}$ settings of the non-deterministic bits in parallel), which cannot compute Parity [28], even if augmented with Mod_p gates for p an odd prime [46, 52, 17]. As GI is DET-hard [55]—and hence can compute Parity—there can be no AC^0 reduction from GI to GpI.

Such a low-depth circuit was quite surprising, although that surprise is perhaps tempered by the use of non-determinism. Nonetheless, it raises the question:

Is it possible that Group Isomorphism is in AC^0?

The authors would be shocked if the answer were "yes," and yet we do not even have results showing that Group Isomorphism cannot be computed by polynomial-size circuits of (!) depth 2. Indeed, it is not clear how to use existing AC^0 lower bound techniques against Group Isomorphism.[3]

In this paper, we aim to close the gap between AC^0 and $\exists^{\log^2 n}\mathsf{FOLL}$ in the complexity of Group Isomorphism and related problems. Our goal is to obtain constant-depth circuits of quasipolynomial size, a natural benchmark in circuit complexity [9]. Significantly improving the size of these circuits would improve the state of the art run-time of Group Isomorphism, a long-standing open question that we do not address here.

Our first main result along these lines is:

THEOREM 1.1. *(Quasi)Group Isomorphism can be solved in* $\mathsf{quasiAC}^0$.

(We discuss quasigroups more below.)

REMARK 1.2. We in fact get a more precise bound of $\exists^{\log^2 n}\mathsf{AC}^0(\mathsf{DTISP}(\mathrm{polylog}(n), \log(n)))$ (Theorem 6.1 gives an even more specific bound), where $\mathsf{DTISP}(t(n), s(n))$ is the class of languages decidable by a Turing machine that simultaneously uses time at most t and space

3 The $\exists^{\log^2 n}\mathsf{FOLL}$ upper bound unconditionally rules out reductions from Parity and Majority. While switching lemmas have been used to get AC^0 lower bounds on LogClique (deciding if a graph has a clique on $O(\log n)$ vertices) [42, 14, 50] (covered in Beame's switching lemma primer [13]), which is in $\exists^{\log^2 n}\mathsf{AC}^0 \subseteq \mathsf{quasiAC}^0$, that problem feels quite different from Group Isomorphism.

at most s. This more precise bound is notable because it is contained in both quasiAC^0 and $\exists^{\log^2 n}\mathsf{FOLL} \cap \exists^{\log^2 n}\mathsf{L}$, the latter thus improving on [17]. We get similarly precise bounds with complicated-looking complexity classes for the other problems mentioned in the introduction, but we omit the precise bounds here for readability.

Focusing on depth bounds—that is, without attempting to improve the worst-case runtime—our result is close to the end of the line for a series of works stretching back to 1970; see Table 1. The only possible further improvements we see, without improving the worst-case runtime, are to get an $\exists^{\log^2 n}\mathsf{AC}^0$ upper bound—for which there are several obstacles, see Section 8—or improving the exact size and depth of our result, but there is not much room for improvement here, as we already get quasi-polynomial-size circuits of depth only 4 and size $n^{O(\log n)}$, matching the current-best serial runtime up to the constant in the exponent (we have not attempted to optimize the constant hidden in the big-Oh; we certainly do not get a constant less than 1, and we conservatively estimate our proof yields a constant not more than 20).

Year	Result	Depth	Citation
1970	Generator-enumerator introduced	$\mathrm{poly}(n)$	Felsch & Neubüser [23][4]
1978	$\exists^{\log^2 n}\mathsf{P} \subseteq \mathsf{DTIME}(n^{\log n + O(1)})$	$\mathrm{poly}(n)$	Tarjan (see Miller [43])
1977	$\mathsf{DSPACE}(\log^2 n)$	$O(\log^2 n)$	Lipton–Snyder–Zalcstein [38][5]
1994	$\exists^{\log^2 n}\mathsf{AC}^1$	$O(\log n)$	Wolf [59][6]
2010	$\exists^{\log^2 n}\mathsf{SAC}^1$	$O(\log n)$	Wagner [58]
2010	$\exists^{\log^2 n}\mathsf{FOLL} \cap \exists^{\log^2 n}\mathsf{L}$	$O(\log\log n)$	Chattopadhyay–Torán–Wagner [17][7]
2013	$\exists^{\log^2 n}\mathsf{SC}^2 \cap \exists^{\log^2 n}\mathsf{L}$	$O(\log n)$	Papakonstantinou–Tang–Qiao [53][8]
2024	$\exists^{\log^2 n}\mathsf{AC}^0(\mathsf{DTISP}(\mathrm{polylog}(n), \log(n))) \subseteq \mathrm{quasi}\mathsf{AC}^0$	4	This work

Table 1. History of the low-level circuit complexity of algorithms for (Quasi)Group Isomorphism based on the generator-enumerator technique. For non-circuit classes, we list their depth as the best-known depth of their simulation by circuits. The class in our bound is contained in all the other classes listed in the table. Although depth 4 in our result does not follow from the complexity class as listed here, it follows from the more exact class we use in Theorem 6.1.

4 Complexity not analyzed there, but the same as Tarjan's algorithm [43].

5 Despite the publication dates, this seems to have been independent of Tarjan's result. They note that Miller and Rabin had also observed this result independently.

6 Wolf only claims a bound of $\exists^{\log^2 n}\mathsf{NC}^2$. However, if we replace his use of NC^1 circuits to multiply two elements of a quasigroup with AC^0 circuits, we immediately get the $\exists^{\log^2 n}\mathsf{AC}^1$ bound.

7 They do not claim the $\exists^{\log^2 n}\mathsf{L}$ bound, but it follows immediately from their algorithm and results.

8 We have written the result this way, despite $\exists^{\log^2 n}\mathsf{L} \subseteq \exists^{\log^2 n}\mathsf{SC}^2$, because in Tang's thesis [53], the only place this is currently published, they only claim NSC^2 using only $O(\log^2 n)$ bits of nondeterminism, which in our notation would be $\exists^{\log^2 n}\mathsf{SC}^2$. However, their algorithm and results also immediately yields a $\exists^{\log^2 n}\mathsf{L}$ bound.

We note that although there are depth reduction techniques for bounded-depth circuits with Mod_p gates [2], no such result is known for $\mathsf{quasiAC}^0$ circuits (without Mod_p gates).

Minimum generating set. Another very natural problem in computational algebra is the Min Generating Set (MGS) problem. Given a group, this problem asks to find a generating set of the smallest possible size. Given that many algorithms on groups depend on the size of a generating set, finding a minimum generating set has the potential to be a widely applicable subroutine. The MGS problem for groups was shown to be in P by Lucchini & Thakkar only very recently [40]. We improve their complexity bound to:

THEOREM 1.3. *Min Generating Set for groups can be solved in* $\mathsf{quasiAC}^0$ *and in* $\mathsf{AC}^1(\mathsf{L})$ *($O(\log n)$-depth, unbounded fan-in circuits with a logspace oracle).*

We note that, although $\mathsf{quasiAC}^0$ is incomparable to P because of the quasi-polynomial size (whereas $\mathsf{AC}^1(\mathsf{L}) \subseteq \mathsf{P}$), the key we are focusing on here is reducing the depth.

For nilpotent groups (widely believed to be the hardest cases of GpI), if we only wish to compute the minimum *number* of generators, we can further improve this complexity to $\mathsf{L} \cap \mathsf{AC}^0(\mathsf{NTISP}(\mathrm{polylog}(n), \log(n)))$ (Proposition 7.3).

While our $\mathsf{AC}^1(\mathsf{L})$ bound above is essentially a careful complexity analysis of the polynomial-time algorithm of Lucchini & Thakkar [40], the $\mathsf{quasiAC}^0$ upper bound is in fact a consequence of our next, more general, result for *quasi*groups, which involves some new ingredients.

Enter quasigroups. Quasigroups can be defined in (at least) two equivalent ways: (1) an algebra whose multiplication table is a Latin square,[9] or (2) a group-like algebra that need not have an identity nor be associative, but in which left and right division are uniquely defined, that is, for all a, b, there are unique x and y such that $ax = b$ and $ya = b$.

In the paper in which they introduced $\log^2(n)$-bounded nondeterminism, Papadimitriou and Yannakakis showed that for arbitrary magmas,[10] testing whether the magma has $\log n$ generators was in fact *complete* for $\exists^{\log^2 n}\mathsf{P}$, and conjectured:

CONJECTURE 1.4 (Papadimitriou & Yannakakis [45, p. 169]). *Min Generating Set for Quasigroups is* $\exists^{\log^2 n}\mathsf{P}$*-complete.*

They explicitly did *not* conjecture the same for MGS for *groups*, writing:

> "We conjecture that this result [$\exists^{\log^2 n}\mathsf{P}$-completeness] also holds for the more structured MINIMUM GENERATOR SET OF A QUASIGROUP problem. In contrast, QUASIGROUP ISOMORPHISM was recently shown to be in DSPACE($\log^2 n$) [59]. Notice that

9 A Latin square is an $n \times n$ matrix where for each row and each column, the elements of $[n]$ appear exactly once.

10 A magma is a set M together with a function $M \times M \to M$ that need not satisfy any additional axioms.

> the corresponding problems for groups were known to be in DSPACE($\log^2 n$) [38].”—Papadimitriou & Yannakakis [45, p. 169]

We thus turn our attention to the analogous problems for quasigroups: MGS for quasigroups, Quasigroup Isomorphism, and the key subroutine, Sub-quasigroup Membership. We note that the $\exists^{\log^2 n}$FOLL upper bound of Chattopadhyay, Torán, and Wagner [17] actually applies to Quasigroup Isomorphism and not just GpI; we perform a careful analysis of their algorithm to put Quasigroup Isomorphism into quasiAC^0 as well.

THEOREM 1.5. *Min Generating Set for Quasigroups is in* quasiAC$^0 \cap$ DSPACE($\log^2 n$).

To the best of our knowledge, MGS for Quasigroups has not been studied from the complexity-theoretic viewpoint previously. While a DSPACE($\log^2 n$) upper bound for MGS for *groups* follows from [53, 3], as far as we know it remained open for quasigroups prior to our work.

As with prior results on Quasigroup Isomorphism and Group Isomorphism [17], and other isomorphism problems such as Latin Square Isotopy and Latin Square Graph Isomorphism [37], Thm. 1.5 implies that Parity does not reduce to MGS for Quasigroups, thus ruling out most known lower bound methods that might be used to prove that MGS for Quasigroups is not in AC^0. We also observe a similar bound for MGS for Groups using Fleischer's technique [25].

Papadimitriou and Yannakakis did not specify the type of reduction used in their conjecture, though their $\exists^{\log^2 n}$P-completeness result for Log Generating Set of a Magma works in both logspace and AC^0 (under a suitable input encoding). Our two upper bounds rule out such reductions for MGS for Quasigroups (unconditionally in one case, conditionally in the other):

COROLLARY 1.6. *The conjecture of Papadimitriou & Yannakakis [45, p. 169] is false under* quasiAC^0 *reductions. It is also false under polylog-space reductions assuming* EXP $\neq$ PSPACE.

In strong contrast, we show that MGS for Magmas is NP-complete (Thm. 7.12).

A key ingredient in our proof of Thm. 1.5 is an improvement in the complexity of another central problem in computational algebra: the Sub-quasigroup Membership problem (Membership,[11] for short):

THEOREM 1.7. *Membership for quasigroups is in* $\exists^{\log^2 n}$DTISP(polylog(n), log(n)) $\subseteq$ quasiAC^0.

Membership for *groups* is well-known to belong to L, by reducing to the connectivity problem on the appropriate Cayley graph (cf. [11, 47]), but as L sits in between AC^0 and AC^1, this is not low enough depth for us.

11 In the literature, the analogous problem for groups is sometimes called Cayley Group Membership or CGM, to highlight that it is in the Cayley table model.

Additional results. We also obtain a number of additional new results on related problems, some of which we highlight here:

— By known AC^0 reductions (see, e.g., Levet [37] for details), our $\mathsf{quasiAC}^0$ analysis of Chattopadhyay, Torán, and Wagner's algorithm for Quasigroup Isomorphism yields $\mathsf{quasiAC}^0$ upper bound for the isomorphism problems for Steiner triple systems, pseudo-STS graphs, Latin square graphs, and Steiner $(t, t+1)$-designs, as well as Latin Square Isotopy. Prior to our work, Quasigroup Isomorphism was not known to be solvable using quasiAC circuits of depth $o(\log\log n)$. See Cor. 6.2.

— Group Isomorphism for simple groups (Cor. 3.3) or for groups from a dense set Y of orders (Thm. 5.1) can be solved in $\mathsf{AC}^0(\mathsf{DTISP}(\mathrm{polylog}(n), \log(n))) \subseteq \mathsf{quasiAC}^0 \cap \mathsf{L} \cap \mathsf{FOLL}$. For groups in a dense set of orders, this improves the parallel complexity compared to the original result of Dietrich & Wilson [22]. As in their paper, note that Y omits large prime powers. Thus, we essentially have that for groups that are *not* p-groups, Group Isomorphism belongs to a *proper* subclass of DET. This evidence fits with the widely-believed idea that p-groups are a bottleneck case for Group Isomorphism.

— Abelian Group Isomorphism (Thm. 4.1) is in $\forall^{\log\log n}\mathsf{MAC}^0(\mathsf{DTISP}(\mathrm{polylog}(n), \log(n)))$. The key novelties here are (1) a new observation that allows us to reduce the number of co-nondeterministic bits from $\log n$ (as in [26]) down to $\log\log n$, and (2) using an $\mathsf{AC}^0(\mathsf{DTISP}(\mathrm{polylog}(n), \log(n)))$ circuit for order finding, rather than FOLL as in [17].

— Membership for nilpotent groups is in $\mathsf{AC}^0(\mathsf{NTISP}(\mathrm{polylog}(n), \log(n))) \subseteq \mathsf{FOLL} \cap \mathsf{quasiAC}^0$ (Prop. 7.3).

1.1 Methods

Several of our results involve careful analysis of the low-level circuit complexity of extant algorithms, showing that they in fact lie in smaller complexity classes than previously known. One important ingredient here is that we use simultaneous time- and space-restricted computations. This not only facilitates several proofs and gives better complexity bounds, but also gives rise to new algorithms such as for Membership for nilpotent groups, which previously was not known to be in FOLL.

One such instance is in our improved bound for order-finding and exponentiation in a semigroup (Lem. 3.1). The previous proof [10] (still state of the art 23 years later) used a then-novel and clever "double-barrelled" recursive approach to compute these in FOLL. In contrast, our proof uses standard repeated doubling, noting that it can be done in $\mathsf{DTISP}(\mathrm{polylog}(n), \log(n)) \subseteq \mathsf{FOLL} \cap \mathsf{quasiAC}^0$, recovering their result with standard tools and reducing the depth; in fact, from our proof we get $\mathsf{quasiAC}^0$ circuits of depth 2, which is clearly optimal from the perspective of depth. We use this improved bound on order-finding to improve the complexity of isomor-

phism testing of Abelian groups (Thm. 4.1), simple groups (Cor. 3.3), and groups of almost all orders (Thm. 5.1).

For a few of our results, however, we need to develop new tools to work with quasigroups. In particular, for the quasiAC^0 upper bound on MGS for quasigroups, we cannot directly adapt the technique of Chattopadhyay, Torán, and Wagner. Indeed, their analysis of their algorithm already seems tight to us in terms of having depth $\Theta(\log\log n)$.

The first key is Thm. 1.7, putting Membership for quasigroups into $\mathsf{NTIME}(\mathrm{polylog}(n))$ (more precisely, into $\exists^{\log^2 n}\mathsf{DTISP}(\mathrm{polylog}(n), \log(n))$). To do this, we replace the use of a cube generating sequences from [17] with something that is nearly as good for the purposes of MGS: we extend the Babai–Szemerédi Reachability Lemma [7, Thm. 3.1] from groups (its original setting) to quasigroups in order to obtain what we call *cube-like* generating sequences, which also give us short straight-line programs. Division in quasigroups is somewhat nuanced, e.g., despite the fact that for any a, b, there exists a unique x such that $ax = b$, this does not necessarily mean that there is an element "a^{-1}" such that $x = a^{-1}b$, because of the lack of associativity. Our proof is thus a careful adaptation of the technique of Babai & Szemerédi, with a few quasigroup twists that result in a slightly worse, but still sufficient, bound.

1.2 Prior work

Isomorphism testing. The best known runtime bound for Gpi is $n^{(1/4)\log_p(n)+O(1)}$-time (where p is the smallest prime dividing n) due to Rosenbaum [49] and Luks [41] (see [36, Sec. 2.2]), though this tells us little about parallel complexity. In addition to Tarjan's result mentioned above [43], Lipton, Snyder, & Zalcstein [38] independently observed that if a group is d-generated, then we can decide isomorphism by considering all possible d-element subsets. This is the *generator enumeration* procedure. Using the fact that every group admits a generating set of size $\leq \log_p(n)$ (where p is the smallest prime dividing n), Tarjan obtained a bound of $n^{\log_p(n)+O(1)}$-time for Group Isomorphism, while Lipton, Snyder, & Zalcstein [38] obtained a stronger bound of $\mathsf{DSPACE}(\log^2 n)$. Miller [43] extended Tarjan's observation to the setting of quasigroups. There has been subsequent work on improving the parallel complexity of generator enumeration for quasigroups, resulting in bounds of $\exists^{\log^2 n}\mathsf{AC}^1$ (AC^1 circuits that additionally receive $O(\log^2 n)$ non-deterministic bits, denoted by other authors as $\beta_2\mathsf{AC}^1$) due to Wolf [59],[12] $\exists^{\log^2 n}\mathsf{SAC}^1$ due to Wagner [58], and $\exists^{\log^2 n}\mathsf{L} \cap \exists^{\log^2 n}\mathsf{FOLL}$ due to Chattopadhyay, Torán, & Wagner [17]. In the special case of groups, generator enumeration is also known to belong to $\exists^{\log^2 n}\mathsf{SC}^2$ [53]. There has been considerable work on polynomial-time, isomorphism tests for several families of groups, as well as more recent work on NC isomorphism tests—we refer to recent works [27, 22, 26] for a survey. We are not aware of work on isomorphism testing for specific families of quasigroups that are not groups.

12 See footnote 6.

Min Generating Set. As every (quasi)group has a generating set of size $\leq \lceil \log n \rceil$, MGS admits an $n^{\log(n)+O(1)}$-time solution for (quasi)groups. In the case of groups, Arvind & Torán [3] improved the complexity to $\mathsf{DSPACE}(\log^2 n)$. They also gave a polynomial-time algorithm in the special case of nilpotent groups. Tang further improved the general bound for MGS for groups to $\exists^{\log^2 n}\mathsf{SC}^2$ [53]. We observe that Wolf's technique for placing Quasigroup Isomorphism into $\mathsf{DSPACE}(\log^2 n)$ also suffices to get MGS for Quasigroups into the same class. Very recently, Das & Thakkar [20] improved the algorithmic upper bound for MGS in the setting of groups to $n^{(1/4)\log(n)+O(1)}$. A month later, Lucchini & Thakkar [40] placed MGS for groups into P. Prior to [40], MGS for Groups was considered comparable to Group Isomorphism in terms of difficulty. Our $\mathsf{AC}^1(\mathsf{L})$ bound (Thm. 1.5) further closes the gap between Membership Testing in groups and MGS for Groups, and in particular suggests that MGS is of comparable difficulty to Membership for groups rather than GpI. Note that Membership for groups has long been known to belong to L [11, 47].

2. Preliminaries

2.1 Algebra and Combinatorics

Graph Theory. A *strongly regular graph* with parameters (n, k, λ, μ) is a simple, undirected k-regular, n-vertex graph $G(V, E)$ where any two adjacent vertices share λ neighbors, and any two non-adjacent vertices share μ neighbors. The complement of a strongly regular graph is also strongly regular, with parameters $(n, n-k-1, n-2-2k+\mu, n-2k+\lambda)$.

A *magma* M is an algebraic structure together with a binary operation $\cdot : M \times M \to M$. We will frequently consider subclasses of magmas, such as groups, quasigroups, and semigroups.

Quasigroups and Latin squares. A *quasigroup* consists of a set G and a binary operation $\star : G \times G \to G$ satisfying the following. For every $a, b \in G$, there exist unique x, y such that $a \star x = b$ and $y \star a = b$. We write $x = a \backslash b$ and $y = b/a$, i. e., $a \star (a \backslash b) = b$ and $(b/a) \star a = b$. When the multiplication operation is understood, we simply write ax for $a \star x$. A *sub-quasigroup* of a quasigroup is a subset that itself is a quasigroup. This means it is closed under the multiplication as well as under left and right quotients. Given $X \subseteq G$, the sub-quasigroup generated by X is denoted as $\langle X \rangle$. It is the smallest sub-quasigroup containing X.

Unless otherwise stated, all quasigroups are assumed to be finite and represented using their Cayley (multiplication) tables.

As quasigroups need not be associative, the parenthesization of a given expression may impact the resulting value. For a sequence $S := (s_0, s_1, \ldots, s_k)$ and parenthesization P from a

quasigroup, define:

$$\mathrm{Cube}(S) = \{P(s_0 s_1^{e_1} \cdots s_k^{e_k}) : e_1, \ldots, e_k \in \{0,1\}\}.$$

We say that S is a *cube generating sequence* if each element g in the quasigroup can be written as $g = P(s_0 s_1^{e_1} \cdots s_k^{e_k})$, for $e_1, \ldots, e_k \in \{0,1\}$. Here, s_i^0 indicates that s_i is not being considered in the product. For every parenthesization, every quasigroup is known to admit a cube generating sequence of size $O(\log n)$ [17].

A *Latin square* of order n is an $n \times n$ matrix L where each cell $L_{ij} \in [n]$, and each element of $[n]$ appears exactly once in a given row or a given column. Latin squares are precisely the Cayley tables corresponding to quasigroups. We will abuse notation by referring to a quasigroup and its multiplication table interchangeably. An *isotopy* of Latin squares L_1 and L_2 is an ordered triple (α, β, γ), where $\alpha, \beta, \gamma : L_1 \to L_2$ are bijections satisfying the following: whenever $ab = c$ in L_1, we have that $\alpha(a)\beta(b) = \gamma(c)$ in L_2. Alternatively, we may view α as a permutation of the rows of L_1, β as a permutation of the columns of L_1, and γ as a permutation of the values in the table. Here, L_1 and L_2 are isotopic precisely if x is the (i, j) entry of L_1 if and only if $\gamma(x)$ is the $(\alpha(i), \beta(j))$ entry of L_2.

Albert showed that a quasigroup Q is isotopic to a group G if and only if Q is isomorphic to G. In general, isotopic quasigroups need not be isomorphic [1].

A Latin square L can equivalently be viewed as a set of triples $\{(i, j, L_{ij}) : i, j \in [n]\} \subseteq [n]\times[n]\times[n]$. Given a set of triples $S \subseteq [n]\times[n]\times[n]$, the Latin square property can equivalently be rephrased as: every $i \in [n]$ appears as the first—resp. second, resp. third—coordinate of some triple in S, and no two triples in S agree in more than one coordinate. From this perspective, an additional potential symmetry of Latin squares emerges: two Latin squares L_1, L_2 are *parastrophic*[13] if there is a permutation $\pi \in S_3$ (where S_3 is the symmetric group of degree 3) such that, when viewed as sets of triples, we have $L_2 = \{(x_{1^\pi}, x_{2^\pi}, x_{3^\pi}) : (x_1, x_2, x_3) \in L_1\}$; the induced map $L_1 \to L_2$ is called a *parastrophy*. A *main class isomorphism* of Latin squares is the composition of a parastrophy and an isotopy; if there exists a main class isomorphism $L_1 \to L_2$, we say they are *main class isomorphic*.[14]

For a given Latin square L of order n, we associate a *Latin square graph* $G(L)$ that has n^2 vertices; one for each triple (a, b, c) that satisfies $ab = c$. Two vertices (a, b, c) and (x, y, z) are adjacent in $G(L)$ precisely if $a = x$ or $b = y$ or $c = z$. Miller showed that two Latin square graphs G_1 and G_2 are isomorphic if and only if the corresponding Latin squares, L_1 and L_2, are main class isomorphic [43].

13 This terminology is borrowed from the quasigroup literature. In the Latin square literature this is sometimes referred to as "conjugate", but we find the argument in Keedwell and Denes [**35, pp. 15–16**] to use the quasigroup nomenclature even in the setting of Latin squares compelling.

14 Miller [**43**], and then Levet [**37**] following Miller, referred to this as "main class isotopy"; we have since found several modern textbooks on quasigroups and Latin squares that refer to this notion as one of "main class isomorphism", "paratopy", or "isostrophy" (sic). Keedwell and Denes [**35, pp. 15–16**] have a nice discussion of the terminology, as well as the history of its usage.

A Latin square graph on n^2 vertices is a strongly regular graph with parameters $(n^2, 3(n-1), n, 6)$. Conversely, a strongly regular graph with these same parameters $(n^2, 3(n-1), n, 6)$ is called a *pseudo-Latin square graph*. Bruck showed that for $n > 23$, a pseudo-Latin square graph is a Latin square graph [16].

Group Theory. For a standard reference, see [48]. All groups are assumed to be finite. For a group G, $d(G)$ denotes the minimum size of a generating set for G. The *Frattini subgroup* $\Phi(G)$ is the set of non-generators of G. If G is a p-group, the Burnside Basis Theorem (see [48, Theorem 5.3.2]) provides that (i) $G = G^p[G,G]$, (ii) $G/\Phi(G) \cong (\mathbb{Z}/p\mathbb{Z})^{d(G)}$, and (iii) if S generates $(\mathbb{Z}/p\mathbb{Z})^{d(G)}$, then any lift of S generates G. A *chief series* of G is an ascending chain $(N_i)_{i=0}^k$ of normal subgroups of G, where $N_0 = 1$, $N_k = G$, and each N_{i+1}/N_i $(i = 0, \ldots, k-1)$ is minimal normal in G/N_i. For $g, h \in G$, the *commutator* $[g,h] := ghg^{-1}h^{-1}$. The *commutator subgroup* $[G,G] = \langle \{[g,h] : g, h \in G\} \rangle$.

Designs. Let $t \leq k \leq v$ and λ be positive integers. A (t,k,λ,v) design is an incidence structure $\mathcal{D} = (X, \mathcal{B}, I)$, where X is a set of v points, $\mathcal{B}$ is a subset of $\binom{X}{k}$—whose elements are referred to as *blocks*—and such that each t-element subset of X belongs to exactly λ blocks. Now I is the point-block incidence matrix, where $I_{x,B} = 1$ precisely if the point x belongs to the block B.

If $t < k < v$, we say that the design is *non-trivial*. If $\lambda = 1$, the design is referred to as a *Steiner design*. We denote Steiner designs as (t,k,v)-designs when we want to specify v the number of points, or Steiner (t,k)-designs when referring to a family of designs. We note that Steiner $(2,3)$-designs are known as *Steiner triple systems*. Projective planes are Steiner $(2, q+1, q^2+q+1)$-designs, and affine planes are Steiner $(2, q, q^2)$-designs. We assume that designs are given by the point-block incidence matrix.

For a design $\mathcal{D} = (X, \mathcal{B}, I)$, we may define a *block intersection graph* (also known as a *line graph*) $G(V,E)$, where $V(G) = \mathcal{B}$ and two blocks B_1, B_2 are adjacent in G if and only if $B_1 \cap B_2 \neq \emptyset$. In the case of a Steiner 2-design, the block-intersection graph is strongly regular. For Steiner triple systems, the block-intersection graphs are strongly regular with parameters $(n(n-1)/6, 3(n-3)/2, (n+3)/2, 9)$. Conversely, strongly regular graphs with the parameters $(n(n-1)/6, 3(n-3)/2, (n+3)/2, 9)$ are referred to as *pseudo-STS graphs*. Bose showed that pseudo-STS graphs with strictly more than 67 vertices are in fact STS graphs [15].

Algorithmic Problems. A multiplication table of a magma $G = \{g_1, \ldots, g_n\}$ of order n is an array M of length n^2 where each entry $M[i + (j-1)n]$ for $i, j \in \{1, \ldots, n\}$ contains the binary representation of k such that $g_i g_j = g_k$. In the following all magmas are given as their multiplication tables.

We will consider the following algorithmic problems. The Quasigroup Isomorphism problem takes as input two quasigroups Q_1, Q_2 given by their multiplication tables, and asks

if there is an isomorphism $\varphi : Q_1 \cong Q_2$. The MEMBERSHIP problem for groups takes as input a group G given by its multiplication table, a set $S \subseteq G$, and an element $x \in G$, and asks if $x \in \langle S \rangle$. We may define the MEMBERSHIP problem analogously when the input is a semigroup or quasigroup, and $\langle S \rangle$ is considered as the sub-semigroup or sub-quasigroup, respectively.

The MINIMUM GENERATING SET (MGS) problem takes as input a magma M given by its multiplication table and asks for a generating set $S \subseteq M$ where $|S|$ is minimum. At some point we will also consider the *decision variant* of MGS: here we additionally give an integer k in the input and the question is whether there exists a generating set of cardinality at most k.

We will be primarily interested in MINIMUM GENERATING SET and MEMBERSHIP in the setting of (quasi)groups, with a few excursions to semigroups and magmas. Note that MEMBERSHIP for groups is known to be in $\mathsf{L} \cap \mathsf{quasiAC}^0$ [11, 47, 25]. Here the containment in L follows from the deep result by Reingold [47] that symmetric logspace (nondeterministic logspace where each transition is also allowed to be applied backward) coincides with L.

2.2 Computational Complexity

We assume that the reader is familiar with standard complexity classes such as L, NL, NP, and EXP. For a standard reference on circuit complexity, see [57]. We consider Boolean circuits using the AND, OR, NOT, and Majority, where $\mathsf{Majority}(x_1, \ldots, x_n) = 1$ if and only if $\geq n/2$ of the inputs are 1. Otherwise, $\mathsf{Majority}(x_1, \ldots, x_n) = 0$. In this paper, we will consider DLOGTIME-uniform circuit families $(C_n)_{n \in \mathbb{N}}$. For this, one encodes the gates of each circuit C_n by bit strings of length $O(\log n)$. Then the circuit family $(C_n)_{n \geq 0}$ is called DLOGTIME-*uniform* if (i) there exists a deterministic Turing machine that computes for a given gate $u \in \{0,1\}^*$ of C_n ($|u| \in O(\log n)$) in time $O(\log n)$ the type of gate u, where the types are $x_1, \ldots, x_n$, NOT, AND, OR, Majority, or oracle gates, and (ii) there exists a deterministic Turing machine that decides for two given gates $u, v \in \{0,1\}^*$ of C_n ($|u|, |v| \in O(\log n)$) and a binary encoded integer i with $O(\log n)$ many bits in time $O(\log n)$ whether u is the i-th input gate for v.

DEFINITION 2.1. Fix $k \geq 0$. We say that a language L belongs to (uniform) NC^k if there exist a (uniform) family of circuits $(C_n)_{n \in \mathbb{N}}$ over the AND, OR, NOT gates such that the following hold:

- The AND and OR gates take exactly 2 inputs. That is, they have fan-in 2.
- C_n has depth $O(\log^k n)$ and uses (has size) $n^{O(1)}$ gates. Here, the implicit constants in the circuit depth and size depend only on L.
- $x \in L$ if and only if $C_{|x|}(x) = 1$.

The complexity class AC^k is defined analogously as NC^k, except that the AND, OR gates are permitted to have unbounded fan-in. That is, a single AND gate can compute an arbitrary conjunction, and a single OR gate can compute an arbitrary disjunction. The class SAC^k is defined analogously, in which the OR gates have unbounded fan-in but the AND gates must have

fan-in 2. The complexity class TC^k is defined analogously as AC^k, except that our circuits are now permitted Majority gates of unbounded fan-in. We also allow circuits to compute functions by using multiple output gates.

Furthermore, for a language L the class $\mathsf{AC}^k(L)$, apart from Boolean gates, also allows oracle gates for L (an oracle gate outputs 1 if and only if its input is in L). If $K \in \mathsf{AC}^k(L)$, then K is said to be AC^k-Turing reducible to L. Finally, for some complexity class C denote $\mathsf{AC}^k(C)$ to be the set of decision problems that are AC^k-Turing reducible to problems in C. Be aware that here we follow the notation of [57], which is different from [58, 26] (where $\mathsf{AC}^k(C)$ is used to denote composition of functions).

For every k, the following containments are well-known:

$$\mathsf{NC}^k \subseteq \mathsf{SAC}^k \subseteq \mathsf{AC}^k \subseteq \mathsf{TC}^k \subseteq \mathsf{NC}^{k+1}.$$

In the case of $k = 0$, we have that:

$$\mathsf{NC}^0 \subsetneq \mathsf{AC}^0 \subsetneq \mathsf{TC}^0 \subseteq \mathsf{NC}^1 \subseteq \mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{SAC}^1 \subseteq \mathsf{AC}^1.$$

We note that functions that are NC^0-computable can only depend on a bounded number of input bits. Thus, NC^0 is unable to compute the AND function. It is a classical result that AC^0 is unable to compute Parity [52]. The containment $\mathsf{TC}^0 \subseteq \mathsf{NC}^1$ (and hence, $\mathsf{TC}^k \subseteq \mathsf{NC}^{k+1}$) follows from the fact that NC^1 can simulate the Majority gate.

We will crucially use the following throughout the paper.

THEOREM 2.2 ([30, Theorem 5.1]). *The product of $(\log n)^{O(1)}$-many integers each of $(\log n)^{O(1)}$ bits can be computed in AC^0.*

We also use the following easy lemma:

LEMMA 2.3. *The prime factors of a $O(\log n)$-bit integer can be computed in AC^0.*

(We remind the reader that we always refer to the uniform classes unless otherwise specified; without uniformity the result would be immediate as all functions of $\log n$ bits are in nonuniform AC^0.)

PROOF. Each prime divisor of n can be represented using $O(\log n)$ bits. As the numbers involved are only $O(\log n)$ bits, ordinary arithmetic function of these numbers can be computed in AC^0, in particular, testing if an $O(\log n)$-bit number is prime, testing if one $O(\log n)$-bit number divides another. So, in parallel, for all numbers $x = 2, 3, \ldots, n/2$, an AC^0 circuit checks which ones are prime and divide n. ■

Further circuit classes. The complexity class MAC^0 is the set of languages decidable by constant-depth uniform circuit families with a polynomial number of AND, OR, and NOT gates,

and a single Majority gate at the output. The class MAC^0 was introduced (but not so named) in [4], where it was shown that $\mathsf{MAC}^0 \subsetneq \mathsf{TC}^0$. This class was subsequently given the name MAC^0 in [32].

The complexity class FOLL is the set of languages decidable by uniform AC circuit families of depth $O(\log \log n)$ and polynomial size. It is known that $\mathsf{AC}^0 \subsetneq \mathsf{FOLL} \subsetneq \mathsf{AC}^1$, and it is open as to whether FOLL is contained in NL [10].

We will be particularly interested in NC circuits of quasipolynomial size (i. e., $2^{O(\log^k n)}$ for some constant k). For a circuit class $C \subseteq \mathsf{NC}$, the analogous class permitting a quasipolynomial number of gates is denoted quasi$C$. We will focus specifically on $\mathsf{quasiAC}^0$. Here, we stress that our results for $\mathsf{quasiAC}^0$ will be stated for the nonuniform setting. Note that DLOGTIME uniformity does not make sense for $\mathsf{quasiAC}^0$, as we cannot encode gate indices using $O(\log n)$ bits. Nonetheless, there exist suitable notions of uniformity for $\mathsf{quasiAC}^0$ [9, 24].

Bounded nondeterminism. For a complexity class C, we define $\exists^{\log^i n} C$ to be the set of languages L such that there exists an $L' \in C$ such that $x \in L$ if and only if there exists y of length at most $O(\log^i |x|)$ such that $(x, y) \in L'$. Similarly, define $\forall^{\log^i n} C$ to be the set of languages L such that there exists an $L' \in C$ such that $x \in L$ if and only if for all y of length at most $O(\log^i |x|)$, $(x, y) \in L'$. For any $i \geq 0$ and any $c \geq 0$, both $\exists^{\log^i n}\mathsf{FOLL}$ and $\forall^{\log^i n}\mathsf{FOLL}$ are contained in quasiFOLL, and so cannot compute Parity [17, 52]. Note that $\forall^{\log n} C \cup \exists^{\log n} C \subseteq \mathsf{AC}^0(C)$.

Time and space-restricted Turing machines. When considering complexity classes defined by Turing machines with a time bound $t(n) \in o(n)$, we use Turing machines with random access and a separate address (or *index*) tape. After writing an address, the machine can go to a query state reading the symbol from the input at the location specified by the address tape. As usual, the machines are otherwise allowed to have multiple work tapes.

For functions $t(n), s(n) \in \Omega(\log n)$, the classes $\mathsf{DTISP}(t(n), s(n))$ and $\mathsf{NTISP}(t(n), s(n))$ are defined to consist of decision problems computable by deterministic (resp. nondeterministic) $t(n)$ time and $s(n)$ space bounded Turing machines. Be aware that there must be one Turing machine that simultaneously satisfies the time and space bound. For details we refer to [57, Section 2.6]. For further reading on the connection to $\mathsf{quasiAC}^0$, we refer to [9, 24].

We frequently use $\mathsf{DTISP}(\mathrm{polylog}(n), \log(n))$ and $\mathsf{NTISP}(\mathrm{polylog}(n), \log(n))$. However, because of how small the time and space bounds are for these classes, when we abuse notation to say some function (not necessarily decision problem) is computable in $\mathsf{DTISP}(\mathrm{polylog}(n), \log(n))$, there is some ambiguity as to what this might mean. We use the following two equivalent definitions. First we need some setup. For a function f from bit-strings to bit-strings, we define its

bit function bit-f as follows. Write $f(x)_i$ to denote the i-th bit of $f(x)$. Then define

$$\text{bit-}f(x,i) = \begin{cases} f(x)_i & i \leq |f(x)| \\ \perp & i > |f(x)|. \end{cases}$$

Note that bit-f is always a function which we may take to have at most two output bits (e.g., by encoding $f(x)_i$ by repeating the bit, and encoding $\perp$ by 01), i.e. a pair of decision problems.

DEFINITION/LEMMA 2.4. *For any function $f\colon \{0,1\}^* \to \{0,1\}^*$ with $|f(x)| \leq O(\log|x|)$, the following are equivalent:*

1. *f is computable by a Turing machine, using $\mathrm{poly}(\log n)$ time and $O(\log n)$ space, that halts with the result written in a specified place on its work tape.*
2. *The two bits of bit-f are each decision problems in $\mathsf{DTISP}(\mathrm{polylog}(n), \log(n))$, i. e., can each be decided by a Turing machine using $\mathrm{poly}(\log n)$ time and $O(\log n)$ space.*

In either case, we say f is computable in $\mathsf{DTISP}(\mathrm{polylog}(n), \log(n))$.

PROOF. ($\Rightarrow$) Suppose f is computable by a Turing machine M using $\mathrm{poly}(\log n)$ time and $O(\log n)$ workspace (*including* the output). Then the following machine N computes the function $(x,i) \mapsto f(x)_i$. On input (x,i), N simulates $M(x)$ while leaving i untouched. After the simulation has completed, one of N's worktapes contains $f(x)_i$, while i is still on the input tape. Next, using another worktape to keep a counter, N walks through $f(x)$ until it gets to the i-th bit, and outputs that bit of $f(x)$.

The entirety of the computation after the simulation of N uses an amount of space that is $|i| \leq O(|f(x)|) \leq O(\log|x|)$, and time that is at most $|i|^2 \leq O(\log^2|x|)$ (in fact, by noting that incrementing a counter by +1 k many times only incurs $O(k)$ many changes to the bits of the counter, this can be reduced to $O(\log|x|)$ as well).

($\Leftarrow$) Suppose the function $(x,i) \mapsto f(x)_i$ is computable by a Turing machine N using $\mathrm{poly}(\log n) = \mathrm{poly}(\log|x|)$ time and $O(\log n) = O(\log|x|)$ space. The following machine M computes f in the manner in which, when started with x on its input tape, at the end of the computation $f(x)$ is the only thing written on the work tape. On input x, M simulates $N(x,0), N(x,1), \ldots$, writing those bits in the specified place on its work tape, until it reaches an i such that $N(x,i) = \perp$. As N was called $|f(x)|+1$ times, this multiplies the time complexity of N by $|f(x)| \leq O(\log|x|)$. However, the space need does not increase except to keep track of the integer i for which the machine will next simulate $N(x,i)$, plus the space for the output, as the space to simulate $N(x,i)$ can be re-used when simulating $N(x,i+1)$. Thus the space used is that of N, plus the space for the output, plus the space to maintain i, which is $|i| \leq \log(|f(x)|+1) \leq O(\log\log|x|)$, so the total space is still $O(\log|x| + \log\log|x|) = O(\log|x|)$. The additional clearing and marking at the end at most doubles the time and adds a constant amount of space usage. ∎

FACT 2.5. $\mathsf{NTISP}(\mathrm{polylog}(n), \log(n)) \subseteq \mathsf{NTIME}(\mathrm{polylog}(n)) \subseteq \mathsf{quasiAC}^0$. *In particular, any decision problem in* $\mathsf{NTIME}(\mathrm{polylog}(n))$ *is computable by a quasi-polynomial-size DNF (a particular kind of depth-2* $\mathsf{quasiAC}^0$ *circuit).*

Furthermore, decision problems in $\mathsf{DTIME}(\mathrm{polylog}(n))$ *are computable by quasi-polynomial-size CNFs as well.*

PROOF. Given a $\mathrm{polylog}(n)$-time nondeterministic Turing machine M, we get a decision tree T_M that decides that same language as follows. Each node of the tree corresponds to any time M either queries an input bit or makes a nondeterministic guess. The two children of such a node correspond to whether the queried (resp., nondeterministically guessed) bit was 0 or 1. The leaves of the tree are labeled accepting or rejecting according to whether M accepts or rejects at that point (note: once no more input bits are queried and no more nondeterministic guesses are made, the answer is fully determined, even though M may deterministically spend an additional $\mathrm{polylog}(n)$ steps figuring out what the answer should be). The decision tree has height at most $\mathrm{polylog}(n)$, and thus at most quasi-polynomially many branches. We get a DNF by taking the disjunction over all accepting paths of the conjunction of the answers to the input queries on those paths.

If there are no non-deterministic guesses made, i.e. our machine M was deterministic, then we can get a CNF from the above decision tree by using De Morgan's law: negate the leaves of the decision tree, get the corresponding DNF of the negated tree, then negate the DNF. (This does not work in the nondeterminstic case, as the disjunction includes all inputs such that there *exists* nondeterministic guesses that lead to an accepting leaf, and when we negate the leaves we end up with co-nondeterminism instead of non-determinism.) ■

LEMMA 2.6. $\mathsf{NTISP}(\mathrm{polylog}(n), \log(n)) \subseteq \mathsf{FOLL}$.

PROOF. This is essentially the proof of Savitch's Theorem: A configuration of a Turing machine consist of the current state, the work and index tape content, but *not* the content of the input tape. For configurations α, β define the Reach predicate as follows:

$$\begin{aligned}
\mathrm{Reach}(\alpha, \beta, 0) &\iff \beta \text{ is reachable from } \alpha \text{ in at most one computation step} \\
\mathrm{Reach}(\alpha, \beta, i) &\iff \exists \gamma : \big(\mathrm{Reach}(\alpha, \gamma, i-1) \wedge \mathrm{Reach}(\gamma, \beta, i-1)\big)
\end{aligned}$$

for $i \geq 1$. Thus, $\mathrm{Reach}(\alpha, \beta, i)$ holds if and only if β can be reached from α in at most 2^i computation steps. Since the running time is bounded by $\log^k n$ for some k, by letting $\mathrm{Start}(w)$ denote the initial configuration for the input $w \in \Sigma^*$ and Accept the accepting configuration of M (which can be assumed to be unique after a suitable manipulation of M), we have

$$\mathrm{Reach}(\mathrm{Start}(w), \mathrm{Accept}, \log\log^k n) \iff w \in L.$$

Now, it remains to observe that the inductive definition of the Reach predicate can be evaluated in FOLL since the recursion depth is $O(\log \log n)$, each recursion step is clearly in AC^0 and there are only polynomially many configurations (because of the space bound of $\log n$). ■

By the very definition we have $\mathsf{DTISP}(\mathrm{polylog}(n), \log(n)) \subseteq \mathsf{L}$ and $\mathsf{NTISP}(\mathrm{polylog}(n), \log(n)) \subseteq \mathsf{NL}$. Thus, we obtain

FACT 2.7.

— $\mathsf{AC}^0(\mathsf{DTISP}(\mathrm{polylog}(n), \log(n))) \subseteq \mathsf{L} \cap \mathsf{FOLL} \cap \mathsf{quasiAC}^0$ *and*

— $\mathsf{AC}^0(\mathsf{NTISP}(\mathrm{polylog}(n), \log(n))) \subseteq \mathsf{NL} \cap \mathsf{FOLL} \cap \mathsf{quasiAC}^0$.

Disjunctive truth-table reductions. We finally recall the notion of a disjunctive truth-table reduction. Again let L_1, L_2 be languages. We say that L_1 is *disjunctive truth-table (dtt)* reducible to L_2, denoted $L_1 \leq_{dtt} L_2$, if there exists a function g mapping a string x to a tuple of strings $(y_1, \ldots, y_k)$ such that $x \in L_1$ if and only if there is some $i \in \{1, \ldots, k\}$ such that $y_i \in L_2$. When g is computable in a complexity class C, we call this a C-dtt reduction, and write $L_1 \leq^C_{dtt} L_2$. For any class C, C-dtt reductions are intermediate in strength between C-many-one reductions and C-Turing reductions.

FACT 2.8. $\exists^{\log^2 n}\mathsf{AC}^0(C)$ *is closed under* $\leq^{\mathsf{AC}^0}_{dtt}$ *reductions for any class* C.

PROOF. Let $L \in \exists^{\log^2 n}\mathsf{AC}^0(C)$, and suppose $L' \leq^{\mathsf{AC}^0}_{dtt} L$ via g. We show that L' is also in $\exists^{\log^2 n}\mathsf{AC}^0(C)$. Let $V \in \mathsf{AC}^0(C)$ be a predicate such that $x \in L \iff [\exists^{\log^2 |x|} w]\, V(x, w)$ for all strings x. Then we have $x \in L'$ iff $\exists\, i \in \{1, \ldots, k\}, \exists\, w \in \{0,1\}^{\log^2 n} : V(g(x)_i, w)$, where $g(x)_i$ denotes the i-th string output by $g(x) = (y_1, \ldots, y_k)$. Since g is, in particular, polynomial size, we have $k \leq \mathrm{poly}(|x|)$, so we have the bit-length of the index i is at most $O(\log |x|)$. Finally, since $\mathsf{AC}^0(C)$ denotes the oracle class, it is closed under AC^0 reductions, and thus the function $(x, w, i) \mapsto V(g(x)_i, w)$ is in $\mathsf{AC}^0(C)$, and therefore $L' \in \exists^{\log^2 n}\mathsf{AC}^0(C)$. ■

3. Order Finding and Applications

In this section, we consider the parallel complexity of order finding. We begin with the following lemma.

LEMMA 3.1. *The following function is in* $\mathsf{DTISP}(\mathrm{polylog}(n), \log(n))$ *(NB: Definition/Lemma 2.4): On input of a multiplication table of a semigroup* S*, an element* $s \in S$*, and a unary or binary number* $k \in \mathbb{N}$ *with* $k \leq |S|$*, compute* s^k.

Note that, because the multiplication table is part of the input, the input size n is always at least $|S|$.

PROOF. If k is given in unary, we first compute its binary representation using a binary search (note that we can write it on the work tape as is uses at most $\lceil \log |S| \rceil$ bits). We identify the semigroup elements with the natural numbers $0, \ldots, |S| - 1$. Now, we compute s^k using the standard fast exponentiation algorithm. Note that the multiplication of two semigroup elements can be done in $\mathsf{DTIME}(\log n)$ as we only need to write down two $\log n$ bit numbers on the address tape (if the multiplication table is not padded up to a power of two, this is still in $\mathsf{DTISP}(\mathrm{polylog}(n), \log(n))$ because we need to multiply two $\log n$ bit numbers to compute the index in the multiplication table).

It is well-known that the fast exponentiation algorithm needs only $O(\log k)$ elementary multiplications and $O(\log k + \log n)$ space; hence, the lemma follows. ■

Note that Lem. 3.1 together with Lem. 2.6 gives a new proof that the problem of computing a power s^k in a semigroup can be done in FOLL. This approach seems easier and more general than the double-barrelled recursive approach in [10].

Lem. 3.1 also yields the following immediate corollary:

COROLLARY 3.2. *On input of a multiplication table of a group G, an element $g \in G$, and $k \in \mathbb{N}$, we may decide whether $\mathrm{ord}(g) = k$ in $\forall^{\log n}\mathsf{DTISP}(\mathrm{polylog}(n), \log(n))$. The same applies if, instead of k, another group element $h \in G$ is given with the question whether $\mathrm{ord}(g) = \mathrm{ord}(h)$. In particular, both questions can be decided by quasi-polynomial-size CNFs (a particular case of depth-2 $\mathsf{quasiAC}^0$).*

PROOF. First check whether $g^k = 1$ using Lem. 3.1. If yes (and $k > 1$), we use $O(\log n)$ universally quantified co-nondeterministic bits to verify that for all $1 \leq i < k$ that $g^i \neq 1$ using again Lem. 3.1. For the second form of input observe that $\mathrm{ord}(g) = \mathrm{ord}(h)$ if and only if for all $i \leq |G|$ we have $g^i = 1$ if and only if $h^i = 1$.

For the depth-2 upper bound: we can compute the decision problems $(g, i) \mapsto [g^i = 1]$, $(g, i) \mapsto [g^i \neq 1]$, and $(g, h, i) \mapsto [g^i = 1 \leftrightarrow h^i = 1]$ in $\mathsf{DTISP}(\mathrm{polylog}(n), \log(n))$, which in turn can be decided by quasi-polynomial-size CNFs. Implementing the above strategy, the $\forall^{\log n}$ becomes an AND gate of fan-in $\mathrm{poly}(n)$. That can be merged with the AND gates at the top of the aforementioned CNFs to get a single CNF of quasi-polynomial size. ■

3.1 Application to isomorphism testing

Using Cor. 3.2, we can improve the upper bound for isomorphism testing of finite simple groups. Previously, this problem was known to be in L [53] and FOLL [26]. We obtain the following improved bound.

COROLLARY 3.3. *Let G be a finite simple group and H be arbitrary. We can decide isomorphism between G and H in $\mathsf{AC}^0(\mathsf{DTISP}(\mathrm{polylog}(n), \log(n)))$*

PROOF. As G is a finite simple group, G is determined up to isomorphism by (i) $|G|$, and (ii) the set of orders of elements of G: $\mathrm{spec}(G) := \{\mathrm{ord}(g) : g \in G\}$ [56]. We may check whether $|G| = |H|$ in AC^0. By Cor. 3.2, we may compute and compare $\mathrm{spec}(G)$ and $\mathrm{spec}(H)$ in $\mathsf{AC}^0(\mathsf{DTISP}(\mathrm{polylog}(n), \log(n)))$. ■

In light of Cor. 3.2, we obtain an improved bound for testing whether a group G is nilpotent. Testing for nilpotency was previously known to be in $\mathsf{L} \cap \mathsf{FOLL}$ [10].

COROLLARY 3.4. *Let G be a finite group given by its multiplication table. We may decide whether G is nilpotent in* $\mathsf{AC}^0(\mathsf{DTISP}(\mathrm{polylog}(n), \log(n)))$.

PROOF. A finite group G is nilpotent if and only if it is the direct product of its Sylow subgroups. As G is given by its multiplication table, we may in AC^0 compute the prime factors of $|G|$ (Lemma 2.3). Thus, for each prime p dividing $|G|$, we identify the elements $X_p := \{g : \mathrm{ord}(g) \text{ is a power of } p\}$ and then test whether X_p forms a group. By Cor. 3.2, we can identify X_p in $\mathsf{AC}^0(\mathsf{DTISP}(\mathrm{polylog}(n), \log(n)))$. Verifying the group axioms is AC^0-computable. The result follows. ■

3.2 Application to membership testing

A group G has the $\log n$ *power basis property* (as defined in [10, 25]) if for every subset $X \subseteq G$ every $g \in \langle X \rangle$ can be written as $g = g_1^{e_1} \cdots g_m^{e_m}$ with $m \leq \log n$ and suitable $g_i \in X$ and $e_i \in \mathbb{Z}$.

OBSERVATION 3.5. *Membership testing for semigroups with the* $\log n$ *power basis property is in* $\mathsf{NTISP}(\mathrm{polylog}(n), \log(n))$.

PROOF. This follows by guessing suitable exponents and using Lem. 3.1 to compute the respective powers. In order to keep the space logarithmically bounded, we do not simply guess $g_1, \ldots, g_m \in X$—which could take up to $\log^2 n$ bits to store—but rather guess the g_i sequentially and only store the running product. That is, first guess g_1 and e_1 using $2 \log n$ bits, and compute $g_1^{e_1}$. Inductively suppose we have computed $g_1^{e_1} \cdots g_i^{e_i}$. Note that we do not need to store the elements $g_1, \ldots, g_i$ and exponents $e_1, \ldots, e_i$, we only keep the product $g_1^{e_1} \cdots g_i^{e_i}$, which takes $\log n$ bits. Then we guess g_{i+1} and e_{i+1}, and continue. ■

This observation allows us to improve several results from [10, 25] from FOLL to $\mathsf{NTISP}(\mathrm{polylog}(n), \log(n))$.

COROLLARY 3.6. *Membership for commutative semigroups is in* $\mathsf{NTISP}(\mathrm{polylog}(n), \log(n))$.

PROOF. Commutative semigroups have the $\log n$ power basis property [25, Lem. 4.2]. ■

COROLLARY 3.7. *Let d be a constant. Membership for solvable groups of class bounded by d is in* $\mathsf{NTISP}(\mathrm{polylog}(n), \log(n))$.

PROOF. Let X be the input set for MEMBERSHIP and G the input group. Theorem 3.5 in [10] is proved by showing that $\langle X\rangle = X_d$ where $X_0 = X$ and for some suitable constant C

$$X_i = \{x_1^{e_1} \cdots x_m^{e_m} \mid m \leq C \log n, x_j \in X_{i-1}, e_j \in \mathbb{Z} \text{ for } j \in \{1, \ldots, m\}\}.$$

Now, as in Observation 3.5, we can guess an element of X_i in $\mathsf{NTISP}(\mathrm{polylog}(n), \log(n))$ given that we have the elements of X_{i-1}. As d is a constant, we can guess the elements of X_{i-1} in $\mathsf{NTISP}(\mathrm{polylog}(n), \log(n))$ by induction. Thus, we can decide membership in $\langle X\rangle = X_d$ in $\mathsf{NTISP}(\mathrm{polylog}(n), \log(n))$. ■

For nilpotent groups, we can do even better and show $\mathsf{NTISP}(\mathrm{polylog}(n), \log(n))$ even without a bound on the nilpotency class—thus, improving considerably over [10]. This also underlines that the class $\mathsf{NTISP}(\mathrm{polylog}(n), \log(n))$ is useful not only because it provides a better complexity bound than FOLL, but it also facilitates some proofs.

THEOREM 3.8. *MEMBERSHIP for nilpotent groups is in* $\mathsf{NTISP}(\mathrm{polylog}(n), \log(n))$.

PROOF. Let X be the input set for MEMBERSHIP and G the input group. Write $n = |G|$. Note that the nilpotency class of G is at most $\log n$. Define $C = \{[x_1, \ldots, x_\ell] \mid x_i \in X, \ell \leq \log n\}$ where $[x_1, \ldots, x_\ell]$ is defined inductively by $[x_1, \ldots, x_\ell] = [[x_1, \ldots, x_{\ell-1}], x_\ell]$ for $\ell \geq 3$. When $\ell = 1$, define $[x_1, \ldots, x_\ell] := x_1$; and when $\ell = 2$, $[x_1, \ldots, x_\ell] := [x_1, x_2]$.

We claim that there is a subset $C' = \{c_1, \ldots, c_m\} \subseteq C$ with $m \leq \log n$ such that every $g \in \langle X\rangle$ can be written as $g = c_1^{e_1} \cdots c_m^{e_m}$ with $e_i \in \mathbb{Z}$.

Let Γ_m denote the m-th term of the lower central series of $\langle X\rangle$ (meaning that $\Gamma_0 = \langle X\rangle$ and $\Gamma_{m+1} = [\Gamma_m, \langle X\rangle]$). Then Γ_m is generated by $C_m = \{[x_1, \ldots, x_k] \mid x_i \in X, k \geq m\}$ (e. g., [18, Lemma 2.6]). Observe that, although k is unbounded, the terms with $k > \log n$ are trivial because $\log n$ is a bound on the nilpotency class. We obtain the desired set C' by first choosing a minimal generating set for the Abelian group Γ_0/Γ_1, then a minimal generating set of Γ_1/Γ_2 and so on (meaning that $(c_1, \ldots, c_m)$ is a so-called *polycyclic generating sequence* of $\langle X\rangle$– see [31, Chapter 8]). Note that m is bounded by $\log n$ since $\langle c_i, \ldots, c_m\rangle$ is a proper subgroup of $\langle c_{i-1}, \ldots, c_m\rangle$ for all i.

Next, let us show that in $\mathsf{NTISP}(\mathrm{polylog}(n), \log(n))$ we can non-deterministically guess elements such that (1) all guessed elements are in $\langle X\rangle$ and (2) every power-product of elements of C' occurs on at least one non-deterministic branch.

This is not difficult: we start by guessing x_1 and set $a = x_1$. Then, guess whether or not to continue; as long as we guess to continue, we guess i and compute $a := [a, x_i]$. As ℓ (in the definition of C) is bounded by $\log n$, we can guess any element of C' in this way in polylogarithmic time. Moreover, note that at any point we only need to store a constant number of group elements. By construction, all such guessed elements are visibly in $\langle X\rangle$.

Once we have guessed an element of C', we can guess a power of it as in Observation 3.5 and then guess the next element of C' and so on. As $m \leq \log n$, this gives an $\mathsf{NTISP}(\mathrm{polylog}(n), \log(n))$ algorithm. ■

4. Abelian Group Isomorphism

In this section, we consider isomorphism testing of Abelian groups. Our main result in this regard is:

THEOREM 4.1. *Let G be an Abelian group, and let H be arbitrary. We can decide isomorphism between G and H in $\forall^{\log\log n}\mathsf{MAC}^0(\mathsf{DTISP}(\mathrm{polylog}(n), \log(n)))$.*

Chattopadhyay, Torán, and Wagner [17] established a $\mathsf{TC}^0(\mathsf{FOLL})$ upper bound on this problem. Grochow & Levet [26, Theorem 5] gave a tighter analysis of their algorithm, placing it in the sub-class $\forall^{\log n}\mathsf{MAC}^0(\mathsf{FOLL})$.[15] We note that Chattopadhyay, Torán, & Wagner also established an upper bound of L for this problem, which is incomparable to the result of Grochow & Levet (*ibid.*). We improve upon both these bounds by (i) showing that $O(\log\log n)$ non-deterministic bits suffice instead of $O(\log n)$ bits, and (ii) using an $\mathsf{AC}^0(\mathsf{DTISP}(\mathrm{polylog}(n), \log(n)))$ circuit for order finding rather than an FOLL circuit. We note that while $\forall^{\log n}\mathsf{MAC}^0(\mathsf{FOLL})$ is contained in $\mathsf{TC}^0(\mathsf{FOLL})$, it is open whether this containment is strict. In contrast, Cor. 4.3 shows that our new bound of $\forall^{\log\log n}\mathsf{MAC}^0(\mathsf{DTISP}(\mathrm{polylog}(n), \log(n)))$ is a class that is in fact *strictly* contained in $\mathsf{L} \cap \mathsf{TC}^0(\mathsf{FOLL})$.

Jeřábek also previously established bounds of $\Sigma_2\text{-}\mathsf{TIME}(\log^2 n)$ for isomorphism testing of Abelian groups, which can be simulated by depth-3 quasiAC^0-circuits of size $n^{O(\log n)}$ [34]; Jeřábek's result and our Theorem 4.1 are incomparable.

PROOF OF THM. 4.1 . Following the strategy of [26, Theorem 7.15], we show that being *non*-isomorphic can be decided in the same class but with existentially quantified non-deterministic bits.

We may check in AC^0 whether a group is Abelian. So if H is not Abelian, we can decide in AC^0 that $G \ncong H$. So suppose now that H is Abelian. By the Fundamental Theorem of Finite Abelian Groups, G and H are isomorphic if and only if their multisets of orders are the same. In particular, if $G \ncong H$, then there exists a prime power p^e such that there are more elements of order p^e in G than in H. We first identify the order of each element, which is $\mathsf{AC}^0(\mathsf{DTISP}(\mathrm{polylog}(n), \log(n)))$-computable by Lem. 3.1.

We will show how to nondeterministically guess and check the prime power p^e such that G has more elements of order p^e than H does. Let $n = p_1^{e_1} \cdots p_\ell^{e_\ell}$ be the prime factorization of n. We have that $\ell \leq \log_2 n$, and the number of distinct prime powers dividing n is $e_1 + \cdots + e_\ell \leq \log_2 n$. Nondeterministically guess a pair (i, e) with $1 \leq i \leq \lfloor \log_2 n \rfloor$ and $1 \leq e \leq \lfloor \log_2 n \rfloor$. We treat this pair as representing the prime power p_i^e at which we will test that G has more elements of that order than H. Because both i and e are bounded in magnitude by $\log_2 n$, the number of bits

15 Grochow & Levet consider $\forall^{\log n}\mathsf{MAC}^0 \circ \mathsf{FOLL}$, where $\circ$ denotes composition (see [26] for a precise formulation). We note that as $\mathsf{AC}^0 \circ \mathsf{FOLL} = \mathsf{FOLL} = \mathsf{AC}^0(\mathsf{FOLL})$, we have that $\forall^{\log n}\mathsf{MAC}^0 \circ \mathsf{FOLL} = \forall^{\log n}\mathsf{MAC}^0(\mathsf{FOLL})$. Thus, Thm. 4.1 improves upon the previous bound of $\forall^{\log n}\mathsf{MAC}^0(\mathsf{FOLL})$ obtained by Grochow & Levet.

guessed is at most $\log_2 \log_2 n$. So we may effectively guess p^e using $O(\log \log n)$ bits to specify p (implicitly, i.e., by its index i) and to specify e (explicitly, i.e., in its binary expansion).

However, to identify elements of order p^e, we will need the number p^e explicitly, in its full binary expansion. First we show that we can get p^e explicitly if we can get p explicitly. Once we have p in its binary expansion, the function $(p, e) \mapsto p^e$ can be computed in AC^0, by Thm. 2.2. Thus all that remains is to get p explicitly.

By Lemma 2.3 we can compute in AC^0 the prime factors of $|G|$. Consider this list of primes as a list of length $O(\log n)$, consisting of numbers each of $O(\log n)$ bits. Now for each pair of primes p_j, p_h, we define an indicator $X(j,h) = 1 \iff p_j > p_h$. As p_j, p_h are representable using $O(\log n)$ bits, we may compute $X(j, \ell)$ in AC^0. Now as the number of primes $\ell \leq \log_2(n)$, we may in AC^0 find a prime j with:

$$\sum_{h=1}^{\ell} X(j,h) = i.$$

The result now follows. ■

As with many of our other results, we show that this class is restrictive enough that it cannot compute Parity. To do this, we appeal to the following theorem of Barrington & Straubing:

THEOREM 4.2 (Barrington & Straubing [12, Thm. 7]). *Let $k > 1$. Any* TC *circuit family of constant depth, size $2^{n^{o(1)}}$, and with at most $n^{o(1)}$* Majority *gates cannot compute the MOD_k function.*

COROLLARY 4.3. *Let $k > 1$, and let $Q^{o(\log n)}$ be any finite sequence (of $O(1)$ length) of alternating $\exists$ and $\forall$ quantifiers, where the total number of bits quantified over is $o(\log n)$. Then*

$$\mathrm{MOD}_k \notin Q^{o(\log n)}\mathsf{MAC}^0(\mathsf{DTISP}(\mathrm{polylog}(n), \log(n))).$$

PROOF. Let $L \in Q^{o(\log n)}\mathsf{MAC}^0(\mathsf{DTISP}(\mathrm{polylog}(n), \log(n)))$. Since by Fact 2.5 we know that $\mathsf{DTISP}(\mathrm{polylog}(n), \log(n)) \subseteq \mathsf{quasiAC}^0$, we have $\mathsf{MAC}^0(\mathsf{DTISP}(\mathrm{polylog}(n), \log(n))) \subseteq \mathsf{quasiMAC}^0$, that is, quasi-polynomial size circuits of constant depth with a single Majority gate at the output. Thus $L \in Q^{o(\log n)}\mathsf{quasiMAC}^0$.

Let C be the $\mathsf{quasiMAC}^0$ circuit such that $x \in L \iff (Qy)C(x,y)$, where $|y| < o(\log |x|)$, for all strings x. There are $2^{o(\log n)} = n^{o(1)}$ possible choices for y; let $C_y(x) = C(x,y)$, where C_y denotes the circuit C with the second inputs fixed to the string y.

Now, to compute L, the quantifiers $Q^{o(\log n)}$ can be replaced by a constant-depth circuit (whose depth is equal to the number of quantifier alternations), and whose total size is $2^{o(\log n)} = n^{o(1)}$, where at each leaf of this circuit, we put the corresponding circuit C_y. The resulting circuit is a $\mathsf{quasiTC}^0$ circuit with only $n^{o(1)}$ Majority gates, hence by Theorem 4.2, cannot compute Parity. ■

5. Isomorphism for Groups of Almost All Orders

Dietrich & Wilson [22] previously established that there exists a dense set $Y \subseteq \mathbb{N}$ such that if $n \in Y$ and G_1, G_2 are magmas of order n given by their multiplication tables, we can (i) decide if G_1, G_2 are groups, and (ii) if so, decide whether $G_1 \cong G_2$ in time $O(n^2 \log^2 n)$, which is quasi-linear time relative to the size of the multiplication table.

In this section, we establish the following.

THEOREM 5.1. *Let $Y \subseteq \mathbb{N}$ be the dense set considered by Dietrich and Wilson [22]. Let $n \in Y$, and let G_1, G_2 be groups of order n. We can decide isomorphism between G_1 and G_2 in $\mathsf{AC}^0(\mathsf{DTISP}(\mathrm{polylog}(n), \log(n)))$.*

Note that verifying the group axioms is AC^0-computable.

REMARK 5.2. Theorem 5.1 provides that for almost all orders, Group Isomorphism belongs to $\mathsf{AC}^0(\mathsf{DTISP}(\mathrm{polylog}(n), \log(n)))$, which is contained within $\mathsf{L} \cap \mathsf{FOLL} \subsetneq \mathsf{P}$ and cannot compute Parity. While it is known that Group Isomorphism belongs to complexity classes such as $\exists^{\log^2 n}\mathsf{L} \cap \exists^{\log^2 n}\mathsf{FOLL}$ [17] and quasiAC^0 (Theorem 6.1) that cannot compute Parity, membership within P—let alone a subclass of P that cannot compute Parity—is a longstanding open problem.

PROOF OF THEOREM 5.1. Dietrich & Wilson showed [22, Theorem 2.5] that if G is a group of order $n \in Y$, then $G = H \ltimes B$, where $\gcd(|B|, |H|) = 1$ and:

- B is a cyclic group of order $p_1 \cdots p_\ell$, where for each $i \in [\ell]$, $p_i > \log\log n$ and p_i is the maximum power of p_i dividing n.
- $|H| = (\log n)^{\mathrm{poly}\log\log n}$; and in particular, if a prime divisor p of n satisfies $p \leq \log\log n$, then p divides $|H|$.

As G_1, G_2 are given by their multiplication tables, we may in AC^0 compute (i) the prime divisors $p_1, \ldots, p_k$ of n, and (ii) determine whether, for each $i \in [k]$, p_i is the maximal power of p_i dividing n. Furthermore, in AC^0, we may write down $\lfloor \log\log n \rfloor$ and test whether $p_i > \lfloor \log\log n \rfloor$.

Fix a group G of order n. We will first discuss how to decompose $G = H \ltimes B$, as prescribed by [22, Theorem 2.5]. Without loss of generality, suppose that $p_1, \ldots, p_\ell$ ($\ell \leq k$) are the unique primes where p_i ($i \in [\ell]$) divides n only once and $p_i > \log\log n$. Now as each p_i can be represented as a string of length $\leq \lceil \log(n) \rceil + 1$, we may in AC^0 compute $\overline{p} := p_1 \cdots p_\ell$ (Thm. 2.2). Using Lem. 3.1, we may in $\mathsf{AC}^0(\mathsf{DTISP}(\mathrm{polylog}(n), \log(n)))$ identify an element $g \in G$ of order $\overline{p}$.

Now in AC^0, we may write down the multiplication table for $H_j \cong G_j/B_j$. As $|H_j| \leq (\log n)^{\mathrm{poly}\log\log n}$, there are $\mathrm{poly}(n)$ possible generating sequences for H_j of length at most $\log |H_j|$. By [7] it actually suffices to consider cube generating sequences for H_j. Now given cube generating sequences $\overline{x_j}$ for H_j, we may by the proof of Theorem 6.1 decide whether

$\overline{x_1} \mapsto \overline{x_2}$ extends to an isomorphism of H_1 and H_2 in $\forall^{\log n}\exists^{\log n}\mathsf{DTISP}(\mathrm{polylog}(n), \log(n)) \subseteq \mathsf{AC}^0(\mathsf{DTISP}(\mathrm{polylog}(n), \log(n)))$. As there are only $\mathrm{poly}(n)$ such generating sequences to consider, we may decide whether $H_1 \cong H_2$ in $\mathsf{AC}^0(\mathsf{DTISP}(\mathrm{polylog}(n), \log(n)))$.

Suppose $H_1 \cong H_2$, $B_1 \cong B_2$, and $\gcd(|B_j|, |H_j|) = 1$ for $j = 1, 2$. We have by the Schur–Zassenhaus Theorem that $G_j = H_j \ltimes_{\theta_j} B_j$ $(j = 1, 2)$ for some action $\theta_j \colon H_j \to \mathrm{Aut}(B_j)$. By Taunt's Lemma [54], it remains to test whether the actions θ_1 and θ_2 are equivalent in the following sense: do there exist isomorphisms $\alpha \colon H_1 \cong H_2$ and $\beta \colon B_1 \cong B_2$ such that

$$\beta(hbh^{-1}) = \alpha(h)\beta(b)\alpha(h^{-1}) \qquad \forall h \in H_1, b \in B_1?$$

Note that, as B_j is Abelian, for any two elements h_1, h_2 of G_j belonging to the same coset of B_j and any element $b \in B_j$, that $h_1 b h_1^{-1} = h_2 b h_2^{-1}$, so the above conjugation action is well-defined, independent of the choice of isomorphic copy of H_j in G_j. Next, since the B_j are cyclic, if b_1 generates B_1, then the above condition is satisfied iff for all $h \in H_1$ we have

$$\beta(hb_1h^{-1}) = \alpha(h)\beta(b_1)\alpha(h^{-1}) \qquad \forall h \in H_1.$$

As above, in $\mathsf{AC}^0(\mathsf{DTISP}(\mathrm{polylog}(n), \log(n)))$ we can find a generator $b_1 \in B_1$ and a cube generating sequence $h_1, \ldots, h_k \in H_1$. In parallel, for all $\mathrm{poly}(n)$ generators $b_2 \in B_2$ and all $\mathrm{poly}(n)$ cube generating sequences $h'_1, \ldots, h'_k \in H_2$, we then test the above condition on the isomorphisms specified by $\beta(b_1) = b_2$ and $\alpha(h_i) = h'_i$ for $i = 1, \ldots, k$. That is, for each choice of b_2 we write down the isomorphism $B_1 \to B_2$ defined by $\beta(b_1) = b_2$, and for each choice of cube generating sequence $h'_1, \ldots, h'_k \in H_2$, we check that

$$\beta(h_i b_1 h_i^{-1}) = h'_i b_2 (h'_i)^{-1} \qquad \forall i = 1, \ldots, k.$$

Checking the preceding condition only involves a constant number of group multiplications, which can thus be done in $\mathsf{DTISP}(\mathrm{polylog}(n), \log(n))$. ∎

6. Quasigroup Isomorphism

In this section, we establish the following.

THEOREM 6.1. *Quasigroup Isomorphism belongs to* $\exists^{\log^2 n}\forall^{\log n}\exists^{\log n}\mathsf{DTISP}(\mathrm{polylog}(n), \log(n))$. *In particular, it can be solved by* $\mathsf{quasiAC}^0$ *circuits of depth 4 and size* $n^{O(\log n)}$.

Note that we have

$$\begin{aligned}\exists^{\log^2 n}\forall^{\log n}\exists^{\log n}\mathsf{DTISP}(\mathrm{polylog}(n), \log(n)) &\subseteq \exists^{\log^2 n}\mathsf{AC}^0(\mathsf{DTISP}(\mathrm{polylog}(n), \log(n))) \\ &\subseteq \mathsf{quasiAC}^0 \cap \exists^{\log^2 n}\mathsf{L} \cap \exists^{\log^2 n}\mathsf{FOLL}.\end{aligned}$$

This statement is inspired by [21] where a similar problem is shown to be in the third level of the polynomial-time hierarchy using the same approach. Our proof follows closely the algorithm for [17, Theorem 3.4].

PROOF OF THEOREM 6.1. Let G and H be quasigroups given as their multiplication tables. We assume that the elements of the quasigroups are indexed by integers $1, \dots, |G|$. If $|G| \neq |H|$ (this can be tested in $\mathsf{DTIME}(\log n)$ by a standard binary search), we know that G and H are not isomorphic. Otherwise, let us write $n = |G|$ and $k = \lceil 2 \log n \rceil + 1$ (while [17, Theorem 3.3] only states that some $k \in O(\log n)$ suffices, the corresponding proof states that for $k = \lceil 2 \log n \rceil + 1$, there is actually a cube generating set).

The basic idea is to guess cube generating sequences $(g_0, \dots, g_k)$ and $(h_0, \dots, h_k)$ for G and H and verify that the map $g_i \mapsto h_i$ induces an isomorphism between G and H. Hence, we start by guessing cube generating sequences $(g_0, \dots, g_k)$ and $(h_0, \dots, h_k)$ with respect to the parenthesization P where the elements are evaluated left-to-right (so $P(g_1 g_2 g_3) = (g_1 g_2) g_3$), with $g_i \in G$, $h_i \in H$. This amounts to guessing $2k \cdot \log(n) \in O(\log^2 n)$ many bits (thus, $\exists^{\log^2 n}$). Now, we need to verify two points in $\forall^{\log n} \exists^{\log n} \mathsf{DTISP}(\mathrm{polylog}(n), \log(n))$:

— that these sequences are actually cube generating sequences,
— that $g_i \mapsto h_i$ induces an isomorphism.

Let us describe the first point for G (for H this follows exactly the same way): we universally verify for every element $g \in G$ (which can be encoded using $O(\log n)$ bits, hence, $\forall^{\log n}$) that we can existentially guess a sequence $(e_1, \dots, e_k) \in \{0,1\}^k$ (i.e. $\exists^{\log n}$) such that $g = P(g_0 g_1^{e_1} \cdots g_k^{e_k})$. We can compute this product in $\mathsf{DTISP}(\log^{2+o(1)} n, \log n)$ by multiplying from left to right:[16] Each multiplication can be done in time $O(\log^{1+o(1)} n)$ because we simply need to compute $i + j \cdot n$ for two addresses i, j of quasigroup elements, write the result on the index tape and then read the corresponding product of group elements from the multiplication table. Moreover, note that for this procedure we only need to store one intermediate result on the working tape at any time , and one quasigroup multiplication only queries $O(\log n)$ bits (this point will be important for our size analysis below); the "$+o(1)$" in the exponent is merely to perform the arithmetic. Thus, computing the product $P(g_0 g_1^{e_1} \cdots g_k^{e_k})$ can be done in time $O(\log^{2+o(1)} n)$ and space $O(\log n)$ and it can be checked whether the result is g.

To check the second point, by [17], we need to verify universally that for all $(c_1, \dots, c_k)$, $(d_1, \dots, d_k)$, $(e_1, \dots, e_k) \in \{0,1\}^k$ (hence, $\forall^{\log n}$) whether

$$\begin{aligned} & P(g_0 g_1^{c_1} \cdots g_k^{c_k}) \cdot P(g_0 g_1^{d_1} \cdots g_k^{d_k}) = P(g_0 g_1^{e_1} \cdots g_k^{e_k}) \\ \iff & P(h_0 h_1^{c_1} \cdots h_k^{c_k}) \cdot P(h_0 h_1^{d_1} \cdots h_k^{d_k}) = P(h_0 h_1^{e_1} \cdots h_k^{e_k}) \end{aligned}$$

16 Here, we could impose the additional requirement that the length of each row/column of the multiplication table is padded up to a power of two in order to get a bound of $\exists^{\log^2 n} \forall^{\log n} \exists^{\log n} \mathsf{DTISP}(\log^2 n, \log n)$).

These products can be computed in the same asymptotic complexity bounds as the product above, using the same technique. Now, it remains to observe that we can combine the two $\forall^{\log n}$ blocks into one $\forall^{\log n}$ block because the check whether $g_i \mapsto h_i$ induces an isomorphism does not depend on the existentially guessed bits in the first check.

The depth 4 circuit bound then follows from the DNF for $\mathsf{DTIME}(\mathrm{polylog}(n))$ (Fact 2.5): the quantifier blocks $\exists^{\log^2 n}\forall^{\log n}\exists^{\log n}$ turn into $\bigvee^{\log^2 n}\bigwedge^{\log n}\bigvee^{\log n}$, and then the disjunction at the top of the DNF for $\mathsf{DTIME}(\mathrm{polylog}(n))$ can be merged into the final $\bigvee$ from the quantifier blocks. The size bound is computed as follows: the $\bigvee^{\log^2 n}\bigwedge^{\log n}\bigvee^{\log n}$ multiplies the size by $2^{O(\log^2 n+2\log n)}$. For each of the above two points, we showed it can be decided in $\mathsf{DTIME}((\log n)^{2+o(1)})$ using at most $O(\log^2 n)$ queries to the input. By the proof of Fact 2.5, those yield decision trees of depth $O((\log n)^2)$, giving a DNF of size at most $n \cdot 2^{O(\log n)^2}$. Multiplying all these factors together gives size $2^{O(\log^2 n)} = n^{O(\log n)}$. ∎

COROLLARY 6.2. *The following problems belong to* $\exists^{\log^2 n}\mathsf{AC}^0(\mathsf{DTISP}(\mathrm{polylog}(n), \log(n)))$.

(a) *Latin Square Isotopy. In particular, it belongs to* $\exists^{\log^2 n}\forall^{\log n}\exists^{\log n}\mathsf{DTISP}(\mathrm{polylog}(n), \log(n))$, *which yields depth* 4 $\mathsf{quasiAC}^0$ *circuits.*

(b) *Isomorphism testing of Steiner triple systems. In particular, this problem belongs to* $\exists^{\log^2 n}\forall^{\log n}\exists^{\log n}\mathsf{NTISP}(\mathrm{polylog}(n), \log(n))$, *which yields depth* 4 $\mathsf{quasiAC}^0$ *circuits.*

(c) *Isomorphism testing of Latin square graphs.*

(d) *Isomorphism testing of Steiner* $(t, t+1)$*-designs.*

(e) *Isomorphism testing of pseudo-STS graphs.*

For Latin Square Isotopy and isomorphism testing of Steiner triple systems, a careful analysis yields depth-4 $\mathsf{quasiAC}^0$ circuits. In contrast, the reductions from [37] for isomorphism testing of Latin square graphs, Steiner $(t, t+1)$-designs, and pseudo-STS graphs all use circuits of depth at least 4. Obtaining reductions of depth less than 4 will likely require new techniques.

PROOF. We proceed as follows.

(a) We note that the reductions outlined in [43, Theorem 2] and [37, Remark 1.6] in fact allow us to determine whether two quasigroups are isotopic, and not just main class isomorphic. Thus, we have an AC^0-computable disjunctive truth-table reduction from Latin Square Isotopy to Quasigroup Isomorphism. This suffices to yield the bound of $\exists^{\log^2 n}\mathsf{AC}^0(\mathsf{DTISP}(\mathrm{polylog}(n), \log(n)))$.

We now turn to establishing the stronger $\exists^{\log^2 n}\forall^{\log n}\exists^{\log n}\mathsf{DTISP}(\mathrm{polylog}(n), \log(n))$ bound. We carefully analyze the $\exists^{\log^2 n}\mathsf{L} \cap \exists^{\log^2 n}\mathsf{FOLL}$ algorithm from [37, Section 3] using cube generating sequences. Our analysis adapts the proof technique of Theorem 6.1. Let Q_1, Q_2 be quasigroups, and let $k \in O(\log n)$. We begin by guessing three cube generating sequences $\bar{a} = (a_0, a_1, \ldots, a_k), \bar{b} = (b_0, b_1, \ldots, b_k), \bar{c} = (c_0, c_1, \ldots, c_k)$ for Q_1, and $\bar{a'} = (a'_0, a'_1, \ldots, a'_k), \bar{b'} = (b'_0, b'_1, \ldots, b'_k), \bar{c'} = (c'_0, c'_1, \ldots, c'_k)$ for Q_2. This requires

$O(\log^2 n)$ non-deterministic bits ($\exists^{\log^2 n}$). From the proof of Theorem 6.1, we may check in $\forall^{\log n}\exists^{\log n}\mathsf{DTISP}(\mathrm{polylog}(n), \log(n))$ whether each of $\bar{a}, \bar{b}, \bar{c}$ generates Q_1, and whether $\bar{a}', \bar{b}', \bar{c}'$ generates Q_2.

So now suppose that each of $\bar{a}, \bar{b}, \bar{c}$ generates Q_1, and each of $\bar{a}', \bar{b}', \bar{c}'$ generates Q_2. Let $\alpha, \beta, \gamma : Q_1 \to Q_2$ be the bijections induced by the respective maps $\bar{a} \mapsto \bar{a}', \bar{b} \mapsto \bar{b}'$, and $\bar{c} \mapsto \bar{c}'$. Now for each pair $(g, h) \in Q_1$, there exist $x, y, z \in \{0,1\}^k$ s.t.:

$$\begin{aligned} g &= a_0 a_1^{x_1} \cdots a_k^{x_k}, \\ h &= b_0 b_1^{y_1} \cdots b_k^{y_k} \\ gh &= c_0 c_1^{z_1} \cdots c_k^{z_k}. \end{aligned}$$

We require $O(\log n)$ universally quantified co-nondeterministic bits ($\forall^{\log n}$) to represent (g, h), and $O(\log n)$ existentially quantified non-deterministic bits to represent x, y, z ($\exists^{\log n}$). Given x, y, z, we may write down g, h, gh in $\mathsf{DTISP}(\mathrm{polylog}(n), \log(n))$ (following the proof of Theorem 6.1). Similarly, let:

$$\begin{aligned} g' &= a'_0 (a'_1)^{x_1} \cdots (a'_k)^{x_k}, \\ h' &= b'_0 (b'_1)^{y_1} \cdots (b'_k)^{y_k} \\ \ell' &= c_0 (c'_1)^{z_1} \cdots (c'_k)^{z_k}. \end{aligned}$$

We can check in $\mathsf{DTISP}(\mathrm{polylog}(n), \log(n))$ whether $g'h' = \ell'$. Similar to the proof of Theorem 6.1, we can combine the two $\forall^{\log n}$ blocks into one $\forall^{\log n}$ block, as the check whether $(\bar{a}, \bar{b}, \bar{c}) \mapsto (\bar{a}', \bar{b}', \bar{c}')$ preserves the quasigroup operation (the *homotopism* condition) is independent of whether each of $\bar{a}, \bar{b}, \bar{c}$ generates Q_1, and each of $\bar{a}', \bar{b}', \bar{c}'$ generates Q_2.

By Fact 2.5, an $\mathsf{NTISP}(\mathrm{polylog}(n), \log(n))$ machine can be simulated by a depth-2 $\mathsf{quasiAC}^0$ circuit, taking care of the final $\exists^{\log n}\mathsf{DTISP}(\mathrm{polylog}(n), \log(n))$. The additional quantifiers $\exists^{\log^2 n}\forall^{\log n}$ then yield depth-4 $\mathsf{quasiAC}^0$ circuits.

(b) Given a Steiner triple system, we may obtain a quasigroup Q in the following manner. For each block $\{a, b, c\}$ in the Steiner triple system, we include the products $ab = c, ba = c, ac = b, ca = b, bc = a, cb = a$. We can write down the multiplication table using an AC^0 circuit, which suffices to obtain a bound of $\exists^{\log^2 n}\mathsf{AC}^0(\mathsf{DTISP}(\mathrm{polylog}(n), \log(n)))$. However, we can use the blocks of the Steiner triple system to look up the relevant products in $\mathsf{NTISP}(\mathrm{polylog}(n), \log(n))$, and so we need not write down the multiplication table. This allows us to directly apply the proof of Theorem 6.1, which yields a bound of

$$\exists^{\log^2 n}\forall^{\log n}\exists^{\log n}\mathsf{NTISP}(\mathrm{polylog}(n), \log(n)),$$

and so we obtain depth-4 quasiAC^0 circuits, as desired. (Analyzing the AC^0 reduction in terms of circuits, yielded only depth 6 in the end, whereas realizing the reduction can be computed in $\mathsf{NTISP}(\mathrm{polylog}(n), \log(n))$ allows us to get depth 4 overall.)

(c) Given two Latin square graphs G_1, G_2, we may recover corresponding Latin squares L_1, L_2 in AC^0 (cf. [37, Lemma 3.9]). Now $G_1 \cong G_2$ if and only if L_1 and L_2 are main class isomorphic [43, Lemma 3]. We may decide whether L_1, L_2 are main class isomorphic using an AC^0-computable disjunctive truth-table reduction to Quasigroup Isomorphism (cf. [37, Remark 1.6]). By Fact 2.8, $\exists^{\log^2 n}\mathsf{AC}^0(\mathsf{DTISP}(\mathrm{polylog}(n), \log(n)))$ is closed under AC^0-computable dtt reductions, thus yielding the bound of $\exists^{\log^2 n}\mathsf{AC}^0(\mathsf{DTISP}(\mathrm{polylog}(n), \log(n)))$.

(d) This immediately follows from the fact that isomorphism testing of Steiner $(t, t+1)$-designs is $\exists^{\log^2 n}\mathsf{AC}^0$-reducible to isomorphism testing of Steiner triple systems [8] (cf., [37, Corollary 4.11]).

(e) Bose [15] previously showed that pseudo-STS graphs with > 67 vertices are STS graphs. Now given a block-incidence graph from a Steiner 2-design with bounded block size, we can recover the underlying design in AC^0 [37, Proposition 4.7]. This yields the desired bound. ■

REMARK 6.3. Latin square graphs are one of the four families of strongly regular graphs under Neumaier's classification [44] (the other families being line graphs of Steiner 2-designs, conference graphs, and graphs whose eigenvalues satisfy the claw bound). Levet [37] previously established an upper bound of $\exists^{\log^2 n}\mathsf{AC}^0$ for isomorphism testing of conference graphs, which is a stronger upper bound than we obtain for Latin square graphs. In contrast, the best known algorithmic runtime for isomorphism testing of conference graphs is $n^{2\log(n)+O(1)}$ due to Babai [6], whereas isomorphism testing of Latin square graphs is known to admit an $n^{\log(n)+O(1)}$-time solution [43].

7. Minimum Generating Set

In this section, we consider the Minimum Generating Set (MGS) problem for quasigroups, as well as arbitrary magmas.

7.1 MGS for Groups in $\mathsf{AC}^1(\mathsf{L})$

In this section, we establish the following.

THEOREM 7.1. *MGS for groups belongs to* $\mathsf{AC}^1(\mathsf{L})$.

We begin with the following lemma.

LEMMA 7.2. *Let G be a group. We can compute a chief series for G in $\mathsf{AC}^1(\mathsf{L})$.*

PROOF. We will first show how to compute the minimal normal subgroups $N_1, \ldots, N_\ell$ of G. We proceed as follows. We first note that the normal closure $\mathrm{ncl}(x)$ is the subgroup generated by $\{gxg^{-1} : g \in G\}$. Now we may write down the elements of $\{gxg^{-1} : g \in G\}$ in AC^0, and then compute $\mathrm{ncl}(x)$ in L using a membership test. Now in L, we may identify the minimal (with respect to inclusion) subgroups amongst those obtained.

Given $N_1, \ldots, N_\ell$, we may easily in L compute $\prod_{i=1}^{k} N_i$ for each $k \leq \ell$. In particular, we may compute $\mathrm{Soc}(G)$ in L. We claim that $\prod_{i=1}^{k} N_i$, for $k = 1, \ldots, \ell$, is in fact a chief series of $\mathrm{Soc}(G)$ (which will then fit into a chief series for G). To see this, we have that $\prod_{i=1}^{k} N_i$ is normal in $\left(\prod_{i=1}^{k} N_i\right) \times N_{k+1}$ and that $(\prod_{i=1}^{k+1} N_i)/(\prod_{i=1}^{k} N_i) \cong N_{k+1}$ is a normal subgroup of $G/\prod_{i=1}^{k} N_i$. By the Lattice Isomorphism Theorem, $(\prod_{i=1}^{k+1} N_i)/(\prod_{i=1}^{k} N_i)$ is in fact minimal normal in $G/\prod_{i=1}^{k} N_i$.

We iterate on this process starting from $G/\mathrm{Soc}(G)$. Note that, as we have computed $\mathrm{Soc}(G)$ from the previous paragraph, we may write down the cosets for $G/\mathrm{Soc}(G)$ in AC^0. Furthermore, given a subgroup $H \leq G/\mathrm{Soc}(G)$, we may write down the elements of $H\mathrm{Soc}(G)$ in AC^0. By the above, the minimal normal subgroups of a group are computable in L. As there are at most $\log n$ terms in a chief series, we may compute a chief series for G in $\mathsf{AC}^1(\mathsf{L})$, as desired. (Recall that we use this notation to mean an AC^1 circuit with oracle gates calling a L oracle, not function composition such as $\mathsf{AC}^1 \circ \mathsf{L}$.) ■

We now prove the Theorem 7.1.

PROOF OF THEOREM 7.1. By Lem. 7.2, we can compute a chief series for G in $\mathsf{AC}^1(\mathsf{L})$. So let $N_1 \triangleleft \cdots \triangleleft N_k$ be a chief series S of G. Lucchini and Thakkar [40] showed that minimum generating sets of G/N_{i+1} have specific structure depending on whether or not N_{i+1}/N_i is Abelian. We proceed inductively down S starting from N_{k-1}. As G/N_{k-1} is a finite simple group, and hence at most 2-generated, we can write all $\binom{n}{2}$ possible generating sets in parallel with a single AC^0 circuit and test whether each generates the group with a membership test. This can be done in L.

Fix $i < k$. Suppose we are given a minimum generating sequence $g_1, \ldots, g_d \in G$ for G/N_i. We will construct a minimum generating sequence for G/N_{i-1} as follows. We consider the following cases:

- **Case 1:** Suppose that $N = N_i/N_{i-1}$ is Abelian. By [39, Theorem 4], we have two cases:
  - **Case 1a:** We have $G/N_{i-1} = \langle g_1, \cdots, g_i, g_j n, g_{j+1}, \cdots, g_d \rangle$ for some $j \in [d]$ and some $n \in N$ (possibly $n = 1$). There are at most $d \cdot |N|$ generating sets to consider in this case and we can test each of them in L.
  - **Case 1b:** If Case 1a does not hold, then we necessarily have that $G/N_{i-1} = \langle g_1, \cdots, g_d, x \rangle$ for any non-identity element $x \in N$.

  Note that there are at most $d \cdot |N| + 1$ generating sets to consider, we may construct a minimum generating set for G/N_{i-1} in L using a membership test.

— **Case 2:** Suppose instead that $N = N_i/N_{i-1}$ is non-Abelian. Then by [40, Corollary 13], the following holds. Let $\eta_G(N)$ denote the number of factors in a chief series with order $|N|$. Let $u = \max\{d, 2\}$ and $t = \min\{u, \lceil \frac{8}{5} + \log_{|N|} \eta_G(N) \rceil\}$. Then there exist $n_1, \ldots, n_t \in N_{i-1}$ (possibly $n_1 = \cdots = n_t = 1$) such that $G/N_{i-1} = \langle g_1 n_1, \cdots, g_t n_t, g_{t+1}, \cdots, g_d \rangle$.
By [40, Corollary 13], there are at most $|N|^{\lceil \frac{8}{5} + \log_{|N|} \eta_G(N) \rceil}$ generating sets of this form. As $\log_{|N|} \eta_G(N) \in O(1)$, we may write down these generating sets in parallel with a single AC^0 circuit and test whether each generate G/N_{i-1} in L using MEMBERSHIP.

Descending along the chief series in this fashion, we compute quotients N_i/N_{i-1} and compute a generating set for G/N_{i-1}. The algorithm terminates when we've computed a generating set for $G/N_0 = G$. Since a chief series has $O(\log n)$ terms, this algorithm requires $O(\log n)$ iterations and each iteration is computable in L. Hence, we have an algorithm for MGS in $\mathsf{AC}^1(\mathsf{L})$. ■

Improving upon the $\mathsf{AC}^1(\mathsf{L})$ bound on MGS for groups appears daunting. It is thus natural to inquire as to families of groups where MGS is solvable in complexity classes contained within $\mathsf{AC}^1(\mathsf{L})$. To this end, we examine the class of nilpotent groups. Arvind & Torán previously established a polynomial-time algorithm for nilpotent groups [3, Theorem 7]. We improve their bound as follows.

PROPOSITION 7.3. *For a nilpotent group G, we can compute $d(G)$ in*

$$\mathsf{L} \cap \mathsf{AC}^0(\mathsf{NTISP}(\mathrm{polylog}(n), \log(n))).$$

PROOF. Let G be our input group. Recall that a finite nilpotent group is the direct product of its Sylow subgroups (which by the Sylow theorems, implies that for a given prime p dividing $|G|$, the Sylow p-subgroup of G is unique). We can, in $\mathsf{AC}^0(\mathsf{DTISP}(\mathrm{polylog}(n), \log(n)))$ (using Cor. 3.2), decide if G is nilpotent; and if so, compute its Sylow subgroups. So we write $G = P_1 \times P_2 \times \cdots \times P_\ell$, where each P_i is the Sylow subgroup of G corresponding to the prime p_i. Arvind & Torán (see the proof of [3, Theorem 7]) established that $d(G) = \max_{1 \le i \le \ell} d(P_i)$. Thus, it suffices to compute $d(P_i)$ for each $i \in [\ell]$.

The Burnside Basis Theorem provides that $\Phi(P) = P^p[P, P]$. We may compute P^p in $\mathsf{L} \cap \mathsf{AC}^0(\mathsf{DTISP}(\mathrm{polylog}(n), \log(n)))$ (the latter using Cor. 3.2). We now turn to computing $[P, P]$. Using a membership test, we can compute $[P, P]$ in L. By [51, I.§4 Exercise 5], every element in $[P, P]$ is the product of at most $\log |P|$ commutators. Therefore, we can also decide membership in $[P, P]$ in $\mathsf{NTISP}(\mathrm{polylog}(n), \log(n))$, and so we can write down the elements of $[P, P]$ in $\mathsf{AC}^0(\mathsf{NTISP}(\mathrm{polylog}(n), \log(n)))$.

Thus, we may compute $\Phi(P)$ in $\mathsf{L} \cap \mathsf{AC}^0(\mathsf{NTISP}(\mathrm{polylog}(n), \log(n)))$. Given $\Phi(P)$, we may compute $|P/\Phi(P)|$ in AC^0. Thus, we may recover $d(P)$ from $|P/\Phi(P)|$ in AC^0, by iterated multiplication of the prime divisor p of $|P|$. As the length of the encoding of p is at most $\log |P|$ and we are multiplying p by itself $\log |P|$ times, iterated multiplication is AC^0-computable. Thus, in total,

we may compute $d(P)$ in $\mathsf{L} \cap \mathsf{AC}^0(\mathsf{NTISP}(\mathrm{polylog}(n), \log(n)))$. It follows that for an arbitrary nilpotent group G, we may compute $d(G)$ in $\mathsf{L} \cap \mathsf{AC}^0(\mathsf{NTISP}(\mathrm{polylog}(n), \log(n)))$. ■

REMARK 7.4. While Prop. 7.3 allows us to compute $d(G)$ for a nilpotent group G, the algorithm is non-constructive. It is not clear how to find such a generating set in L. We can, however, compute such a generating set in $\mathsf{AC}^1(\mathsf{NTISP}(\mathrm{polylog}(n), \log(n)))$. Note that this bound is incomparable to $\mathsf{AC}^1(\mathsf{L})$. We outline the algorithm here.

The Burnside Basis Theorem provides that for a nilpotent group G, (i) every generating set of G projects to a generating set of $G/\Phi(G)$, and (ii) for every generating set S of $G/\Phi(G)$, *every* lift of S is a generating set of G. Furthermore, every minimum generating set of G can be obtained from the Sylow subgroups in the following manner. Write $G = P_1 \times \cdots \times P_\ell$, where P_i is the Sylow p_i-subgroup of G. Suppose that $P_i = \langle g_{i1}, \ldots, g_{ik} \rangle$ (where we may have $g_{ij} = 1$ for certain values of j). Write $g_j = \prod_{i=1}^{\ell} g_{ij}$. As in the proof of [3, Theorem 7] we obtain $G = \langle g_1, \ldots, g_k \rangle$.

Given generating sets for $P_1, \ldots, P_\ell$, we may in $\mathsf{L} \cap \mathsf{AC}^0(\mathsf{NTISP}(\mathrm{polylog}(n), \log(n)))$ recover a generating set for G. Thus, it suffices to compute a minimum generating set for $P/\Phi(P) \cong (\mathbb{Z}/p\mathbb{Z})^{d(P)}$, where P is a p-group. Note that we may handle each Sylow subgroup of G in parallel. To compute a minimum generating set of $P/\Phi(P)$, we use the generator enumeration strategy. As $P/\Phi(P)$ is Abelian, we may check in $\mathsf{AC}^0(\mathsf{NTISP}(\mathrm{polylog}(n), \log(n)))$ (by Cor. 3.6) whether a set of elements generates the group. As $d(P) \leq \log |P|$, we have $\log |P|$ steps where each step is $\mathsf{AC}^0(\mathsf{NTISP}(\mathrm{polylog}(n), \log(n)))$-computable. Thus, we may compute a minimum generating set for P in $\mathsf{AC}^1(\mathsf{NTISP}(\mathrm{polylog}(n), \log(n)))$, as desired.

7.2 MGS for Quasigroups

In this section, we consider the Minimum Generating Set problem for quasigroups. Our goal is to establish the following.

THEOREM 7.5. *For MGS for quasigroups,*

1. *The decision version belongs to* $\exists^{\log^2 n} \forall^{\log n} \exists^{\log n} \mathsf{DTISP}(\mathrm{polylog}(n), \log(n)) \subseteq \mathsf{DSPACE}(\log^2 n)$*;*
2. *The search version belongs to* $\mathsf{quasiAC}^0 \cap \mathsf{DSPACE}(\log^2 n)$*.*

In the paper in which they introduced (polylog-)limited nondeterminism, Papadimitriou and Yannakakis conjectured that MGS for quasigroups was $\exists^{\log^2 n}\mathsf{P}$-complete [45, after Thm. 7]. While they did not specify the type of reductions used, it may be natural to consider polynomial-time many-one reductions. Theorem 7.5 refutes two versions of their conjecture under other kinds of reductions, that are incomparable to polynomial-time many-one reductions: $\mathsf{quasiAC}^0$ reductions unconditionally and polylog-space reductions conditionally. We note that their other $\exists^{\log^2 n}\mathsf{P}$-completeness result in the same section produces a reduction that in fact can be done in logspace and (with a suitable, but natural, encoding of the gates in a circuit) also in AC^0, so our result rules out any such reduction for MGS. (We also note: assuming EXP $\neq$

PSPACE, showing that this problem is complete under polynomial-time reductions would give a separation between poly-time and log-space reductions, an open problem akin to P ≠ L.)

COROLLARY 7.6. *MGS for quasigroups and* Quasigroup Isomorphism *are*

1. *not* $\exists^{\log^2 n}$P*-complete under* quasiAC$^0[p]$ *Turing reductions, or even such reductions up to depth* $o(\log n/\log\log n)$.
2. *not* $\exists^{\log^2 n}$P*-complete under polylog-space Turing reductions unless* EXP = PSPACE.

PROOF. (a) Thm. 7.5 (1) for MGS, resp. Thm. 6.1 for Quasigroup Isomorphism, place both problems into classes that are contained in quasiAC^0 by Fact 2.5. If either problem were $\exists^{\log^2 n}$P-complete under quasi-polynomial size, $o(\log n/\log\log n)$-depth, AC$[p]$ circuits, then, since Parity $\in$ P $\subseteq \exists^{\log^2 n}$P, we would get such circuits for Parity, which do not exist [46, 52] (Smolensky's argument yields a minimum size of $\exp(\Omega(n^{1/(2d)}))$ for depth-d circuits, which is super-polynomial when $d \in o(\log n/\log\log n)$).

(b) Both MGS for quasigroups and Quasigroup Isomorphism are in DSPACE$(\log^2 n)$ by Thm. 7.5, resp. [17]. The closure of DSPACE$(\log^2 n)$ under poly-log space reductions is contained in polyL $= \bigcup_{k\geq 0}$ DSPACE$(\log^k n)$. If either of these two quasigroup problems were complete for $\exists^{\log^2 n}$P under polylog-space Turing reductions, we would get $\exists^{\log^2 n}$P $\subseteq$ polyL. Under the latter assumption, by a straightforward padding argument, we now show that EXP = PSPACE.

Let $L \in$ EXP; let k be such that $L \in$ DTIME$(2^{n^k}+k)$. Define $L_{pad} = \{(x, 1^{2^{|x|^k}+k}) : x \in L\}$. By construction, $L_{pad} \in$ P. Let us use N to denote the size of the input to L_{pad}, that is, $N = 2^{n^k}+k+n$. By assumption, we thus have $L_{pad} \in$ polyL. Suppose ℓ is such that $L_{pad} \in$ DSPACE$(\log^\ell N)$. We now give a PSPACE algorithm for L. In order to stay within polynomial space, we cannot write out the padding $1^{2^{n^k}+k}$ explicitly. What we do instead is simulate the DSPACE$(\log^\ell N)$ algorithm for L_{pad} as follows. Whenever the head on the input tape would move off the x and into the padding, we keep track of its index into the padding, and the simulation responds *as though* the tape head were reading a 1. When the tape head moves right the index increases by 1, when it moves left it decreases by 1, and if the index is zero and the tape head moves left, then we move the tape head onto the right end of the string x. The index itself is a number between 0 and $2^{n^k}+k$, so can be stored using only $O(n^k)$ bits. The remainder of the L_{pad} algorithm uses only $O(\log^\ell N) = O(n^{k\ell})$ additional space, thus this entire algorithm uses only a polynomial amount of space, so $L \in$ PSPACE, and thus EXP = PSPACE. ■

Now we return to establishing the main result of this section, Thm. 7.5. To establish Thm. 7.5 (1) and (2), we will crucially leverage the Membership for quasigroups problem. To this end, we will first establish the following.

THEOREM 7.7. *Membership for quasigroups belongs to* $\exists^{\log^2 n}$DTISP$(\text{polylog}(n), \log(n))$.

Thm. 7.7 immediately yields the following corollary.

COROLLARY 7.8. *For quasigroups, MEMBERSHIP and MGS are not hard under* AC^0*-reductions for any complexity class containing PARITY.*

The Reachability Lemma and proofs. The proofs of Theorem 7.7 and Theorem 7.5 rely crucially on the following adaption of the Babai–Szemerédi Reachability Lemma [7, Theorem 3.1] to quasigroups. We first generalize the notion of a straight-line program for groups [7] to SLPs for quasigroups. We follow the same strategy as in the proof of [7, Theorem 3.1], but there are some subtle, yet crucial, modifications due to the fact that quasigroups are non-associative and need not possess an identity element.

Let X be a set of generators for a quasigroup G. We call a sequence of elements $g_1, \dots, g_\ell \in G$ a *straight-line program* (SLP for short) if each g_i ($i \in [\ell]$) either belongs to X, or is of the form or $g_j g_k$, $g_j \backslash g_k$, or g_j / g_k for some $j, k < i$ (where $g_j \backslash g_k$, resp. g_j / g_k, denotes the quasigroup division as defined in Section 2.1). An SLP is said to *compute* or *generate* a set S (or an element g) if $S \subseteq \{g_1, \dots, g_\ell\}$ (resp. $g \in \{g_1, \dots, g_\ell\}$).

For any sequence of elements $z_1, \dots, z_k$, let $P(z_1 z_2 \cdots z_k)$ denote the left-to-right parenthesization, e.g., $P(z_1 z_2 z_3) = (z_1 z_2) z_3$. For some initial segment $z_0, z_1, \dots, z_i$ define the cube

$$K(i) = \{P(z_0 z_1^{e_1} \cdots z_i^{e_i}) : e_1, \dots, e_i \in \{0, 1\}\},$$

where $e_j = 0$ denotes omitting z_j from the product (since there need not be an identity element). Define $L(i) = K(i) \backslash K(i) = \{g \backslash h : g, h \in K(i)\}$.

Note that, if $L(k) = G$, then $z_1, \dots, z_k$ is very similar to a cube generating sequence and we call it a *cube-like* generating sequence (the difference is that $L(k)$ allows quotients of cube words, not only cube words, and in a quasigroup quotients of cube words need not always be cube words).

LEMMA 7.9 (Reachability Lemma for quasigroups). *Let G be a finite quasigroup and let X be a set of generators for G. Then there exists a sequence $z_0, \dots, z_t$ with $t \leq \log |G|$ such that:*

(1) $L(t) = G$

(2) for each i we have either $z_i \in X$ or $z_i \in \{gh,\ g/h,\ h\backslash g \mid g, h \in L(i-1)\}$.

In particular, for each $g \in G$, there exists a straight-line program over X generating g which has length $O(\log^2 |G|)$.

PROOF. We will inductively construct the sequence $z_0, z_1, \dots, z_t$ as described in the lemma. To start, we take z_0 as an arbitrary element from X. Hence, $K(0) = \{z_0\}$. Next, let us construct $K(i+1)$ from $K(i)$. If $L(i) \neq G$, we set z_{i+1} to be some element of $G \setminus L(i)$ that we can find as follows: As $G \neq L(i)$, either $X \not\subseteq L(i)$ or $L(i)$ is not a sub-quasigroup. Hence, we have one of the following cases:

— If there is some $g \in X \setminus L(i)$, we simply take $z_{i+1} = g$.

— Otherwise, $L(i)$ is not a sub-quasigroup; hence, there exist $g, h \in L(i)$ with one of gh, g/h, or $g\backslash h \notin L(i)$. We choose z_{i+1} to be one of gh, g/h, or $g\backslash h$ that is not in $L(i)$.

Next, we first claim that $|K(i+1)| = 2 \cdot |K(i)|$. Note that $K(i+1) = K(i) \cup K(i)z_{i+1}$ by definition. As right-multiplication by a fixed element is a bijection in a quasi-group, it suffices to show that $K(i) \cap K(i)z_{i+1} = \emptyset$. So, suppose that there exists some $a \in K(i) \cap K(i)z_{i+1}$. Then $a = gz_{i+1}$ for some $g \in K(i)$. Hence, $z_{i+1} = g\backslash a$, contradicting $z_{i+1} \notin L(i)$, since both a and g are in $K(i)$.

It now follows that $|K(i)| = 2^i$ and, hence, for some $t \leq \lceil \log_2(|G|) \rceil$ we have $L(t) = G$. This completes the proof of the first part of the lemma.

It remains to see that for every $g \in G$ there is a straight-line program over X of length $O(\log^2 |G|)$. To see this, let $c(i)$ denote the straight-line cost for $\{z_0, z_1, \ldots, z_i\}$ $(1 \leq i \leq t)$, which is defined as the length of the shortest SLP generating $\{z_0, z_1, \ldots, z_i\}$. We have $c(0) = 1$ and, if $z_{i+1} \in X \setminus L(i)$, then $c(i+1) \leq c(i) + 1$. If $z_{i+1} \in \{gh,\ g/h,\ h\backslash g \mid g, h \in L(i)\}$, we can write each of g and h as SLP over the set $\{z_0, z_1, \ldots, z_i\}$ of length at most $2i+1$, yielding an SLP for z_{i+1} over the same set of length at most $4i+3$. Together this yields $c(i+1) \leq c(i) + 4i + 3$. Hence, by induction, $c(t) \in O(\log^2 |G|)$. As $L(t) = G$, we obtain for any element $g \in G$ an SLP of length $c(t) + 2t + 1 \in O(\log^2 |G|)$. ∎

REMARK 7.10. The proof of Lemma 7.9 shows that for any quasigroup Q and any generating set $S \subseteq Q$, every element $g \in Q$ can be realized with a parse tree of depth $O(\log^2 |Q|)$. In contrast, Wolf [59] establishes the existence of parse trees of depth $O(\log |Q|)$.

For proving Theorem 7.7 we follow essentially the ideas of [25] (though we avoid introducing the notion of Cayley circuits); however, we have to use a more refined approach by not simply guessing an SLP but the complete information given in Lemma 7.9. Fleischer obtained a quasiAC^0 bound for MEMBERSHIP for groups by then showing that the Cayley circuits for this problem can be simulated by a depth-2 quasiAC^0 circuit. We will instead directly analyze the straight-line programs using an $\exists^{\log^2 n}\mathsf{DTISP}(\mathrm{polylog}(n), \log(n))$ (sequential) algorithm.

PROOF OF THEOREM 7.7. To decide whether $g \in \langle X \rangle$ with $X = \{x_1, \ldots, x_\ell\}$, we guess a sequence of elements $z_1, \ldots, z_k \in G$ with $k \leq \log n$ as in Lemma 7.9 together with the necessary information to verify condition (2) in the lemma and whether $g \in L(k)$. This information consists of sequences $\ell_i \in [\ell]$ and $e_{i,j}^{(\mu)} \in \{0,1\}$ for $i \in [k]$, $j \in [i-1]$, $\mu \in [4]$ and, as witness that $g \in L(k)$, a sequence $f_j^{(\mu)} \in \{0,1\}$ for $j \in [k]$, $\mu \in [2]$. Note that all this information can be represented using $O(\log^2 n)$ bits.

In the $\mathsf{DTISP}(\mathrm{polylog}(n), \log(n))$ computation we compute for all i the elements $g_i^{(\mu)} = P(z_0 z_1^{e_{i,1}^{(\mu)}} \cdots z_{i-1}^{e_{i,i-1}^{(\mu)}})$ and verify that z_i is either x_{ℓ_i} or can be written as ab, $a\backslash b$ or a/b where $a = g_i^{(1)}\backslash g_i^{(2)}$ and $b = g_i^{(3)}\backslash g_i^{(4)}$ – thus, verifying the conditions of Lemma 7.9. Finally, with the same

technique we verify that $g \in L(k)$ by checking whether $g = P(z_0 z_1^{f_1^{(1)}} \cdots z_k^{f_k^{(1)}}) \backslash P(z_0 z_1^{f_1^{(2)}} \cdots z_k^{f_k^{(2)}})$. ■

PROOF OF THEOREM 7.5. (1) Let G denote the input quasigroup (of order n). First, every quasigroup has a generating set of size $\leq \lceil \log n \rceil$ [43]. Therefore, we start by guessing a subset $X \subseteq G$ of size at most $\leq \lceil \log n \rceil$ (resp. the size bound given in the input). For this, we use $O(\log^2 n)$ existentially quantified non-deterministic bits ($\exists^{\log^2 n}$). Furthermore, we also guess some additional information, namely, the sequence $z_1, \ldots, z_k \in G$ from Lemma 7.9 and, just as in the proof of Theorem 7.7, the witnesses that this is, indeed, a cube-like sequence. As in the proof of Theorem 7.7 this takes again $O(\log^2 n)$ existentially quantified bits.

In the next step, we verify whether X actually generates G. This is done by checking for all $g \in G$ (universally quantifying $O(\log n)$ bits, $\forall^{\log n}$) whether $g \in \langle X \rangle$, which can be done in $\exists^{\log n}\mathsf{DTISP}(\mathrm{polylog}(n), \log(n)) \subseteq \mathsf{quasiAC}^0$ just as in Theorem 7.7. Here it is crucial to note that, as we have already guessed the cube-like generating set in the outer existential block, it suffices to guess $O(\log n)$ bits for the exponents. Finally, as in the proof of Theorem 7.7, we verify in $\mathsf{DTISP}(\mathrm{polylog}(n), \log(n))$ using the additional information guessed above that $z_1, \ldots, z_k \in G$ is indeed, a cube-like sequence. This concludes the proof of the bound (a) for the decision variant.

(2) To find a minimum-sized generating sequence, we can enumerate all possible generating sets X of size at most $\log n$ in $\mathsf{quasiAC}^0 \cap \mathsf{DSPACE}(\log^2 n)$. If we want to compute the minimum generating set, we first, have to find a generating set X and then we have to check whether X is actually of smallest possible size. To do this, in a last step, we use the decision variant from above to check that there is no generating set of size at most $|X| - 1$ for G. ■

REMARK 7.11. While it is possible to directly reduce MGS to MEMBERSHIP in $\mathsf{quasiAC}^0$, we obtain slightly better bounds by instead directly leveraging the Reachability Lemma 7.9. It is also possible to $\mathsf{quasiAC}^0$-reduce QUASIGROUP ISOMORPHISM to MEMBERSHIP. This formalizes the intuition that, from the perspective of $\mathsf{quasiAC}^0$, membership testing is an essential subroutine for isomorphism testing and MGS.

This might seem surprising, as in the setting of groups, MEMBERSHIP belongs to L, while MGS belongs to $\mathsf{AC}^1(\mathsf{L})$ (Thm. 7.1), yet it is a longstanding open problem whether GROUP ISOMORPHISM is in P.

7.3 MGS for Magmas

In this section, we establish the following.

THEOREM 7.12. *The decision variant of MINIMUM GENERATING SET for commutative magmas (unital or not) is* NP*-complete under many-one* AC^0 *reductions.*

This NP-completeness result helps explain the use of Integer Linear Programming in practical heuristic algorithms for the search version of this problem in magmas, e. g., [33].

For the closely related problem they called Log Generators—given the multiplication table of a binary function (=magma) of order n, decide whether it has a generating set of size $\leq \lceil \log_2 n \rceil$—Papadimitriou and Yannakakis proved that Log Generators of Magmas is $\exists^{\log^2 n}$P-complete under polynomial-time reductions [45, Thm. 7]. Our proof uses a generating set of size roughly $\sqrt{n}$, where n is the order of the magma; this is analogous to the situation that Log-Clique is in $\exists^{\log^2 n}$P, while finding cliques of size $\Theta(\sqrt{n})$ in an n-vertex graph is NP-complete.

PROOF. It is straightforward to see that the problem is in NP by simply guessing a suitable generating set. To show NP-hardness we reduce 3SAT to Minimum Generating Set for commutative magmas. Let $F = \bigwedge_{j=1}^{m} C_j$ with variables $X_1, \ldots, X_n$ and clauses $C_1, \ldots, C_m$ be an instance of 3SAT. Our magma M consists of the following elements:

- for each variable X_i, two elements $X_i, \overline{X}_i$,
- for each clause C_j an element C_j,
- for each $1 \leq j \leq k \leq m$ an element $S_{j,k}$, and
- a trash element 0.

We use S as an abbreviation for $S_{1,m}$.

We define the multiplication as follows:

$$\begin{aligned} C_j X &= S_{j,j} \text{ if the literal } X \text{ appears in } C_j \\ S X_i &= \overline{X}_i, \\ S \overline{X}_i &= X_i, \\ S_{j,k} S_{k+1,\ell} &= S_{j,\ell}. \end{aligned}$$

Aside from multiplication being commutative (e. g., we also have $X_i S = \overline{X}_i$, etc.), all other products are defined as 0.

The idea is that the presence of $S_{j,k}$ in a word indicates that clauses $j, j+1, \ldots, k$ have been satisfied. This interpretation aligns with the multiplication above: viz. $C_j X = S_{j,j}$ if X satisfies C_j. If clauses j through k are satisfied ($S_{j,k}$) and clauses $k+1$ through ℓ are satisfied ($S_{k+1,\ell}$), then in fact clauses j through ℓ are satisfied ($S_{j,k} S_{k+1,\ell} = S_{j,\ell}$). We use "$S$" for all of these as a mnemonic for "satisfied" and also because $S = S_{1,m}$ acts as a "swap" on literals.

Note that any generating set for M must include all the C_j, since without them, there is no way to generate them from any other elements. Similarly, any generating set must include, for each $i = 1, \ldots, n$, at least one of X_i or $\overline{X}_i$, since again, without them, there is no way to generate those from any other elements.

When F is satisfiable, we claim that M can be generated by precisely $n + m$ elements. Namely, include all C_j in the generating set. Fix a satisfying assignment φ to F. If $\varphi(X_i) = 1$, then include X_i in the generating set, and if $\varphi(X_i) = 0$, include $\overline{X}_i$ in the generating set. Since φ is a satisfying assignment, for each j, there is a literal X in our generating set that appears in C_j, and from those two we can generate $C_j X = S_{j,j}$. Next, $S_{j,j} S_{j+1,j+1} = S_{j,j+1}$, and by induction we

can generate all $S_{j,k}$. In particular, we can generate $S = S_{1,m}$, and then using S and our literal generators, we can generate the remaining literals.

Conversely, suppose M is generated by $n+m$ elements; we will show that F is satisfiable. As argued above, those elements must consist of $\{C_j : j \in [m]\}$ together with precisely one literal corresponding to each variable. Since the final defining relation can only produce elements $S_{j,\ell}$ with j strictly less than ℓ, the only way to generate $S_{j,j}$ from our generating set is using the first relation. Namely, at least one of the literals in our generating set must appear in C_j. But then, reversing the construction of the previous paragraph, the literals in our generating set give a satisfying assignment to F.

Identity elements. We may add a new element e, relations $ea = ae = a$ for all a, and include e as a generator. When this is done, the reduction queries whether the algebra is generated by $n+m+1$ elements or more than that many, since the element e must be contained in every generating set. ■

As with most NP-complete decision versions of optimization problems, we expect that the *exact* version—given a magma M and an integer k, decide whether the minimum generating set has size exactly k—is DP-complete, but we leave that as a (minor) open question.

8. Conclusion

The biggest open question about constant-depth complexity on algebras given by multiplication tables is, in our opinion, still whether or not Group Isomorphism is in AC^0 in the Cayley table model. Our results make salient some more specific, and perhaps more approachable, open questions that we now highlight.

First, sticking to the generator-enumerator technique, it would be interesting if the complexity of any single part of our Theorem 6.1 could be improved, even if such an improvement does not improve the complexity of the overall algorithm. Enumerating generators certainly needs $\log^2 n$ bits. Can one verify that a given list of elements is a cube generating sequence better than $\forall^{\log n}\exists^{\log n}\mathsf{DTISP}(\mathrm{polylog}(n), \log(n))$? Can one verify that a given mapping $g_i \mapsto h_i$ of generators induces an isomorphism with complexity lower than $\forall^{\log n}\mathsf{DTISP}(\mathrm{polylog}(n), \log(n))$?

More strongly, is GpI in $\exists^{\log^2 n}\mathsf{AC}^0$? We note that unlike the question of AC^0, a positive answer to this question does not entail resolving whether GpI is in P.

QUESTION 8.1. Does MGS for groups belong to L?

QUESTION 8.2. Does Membership for quasigroups belong to L?

The analogous result is known for groups, by reducing to the connectivity problem on Cayley graphs. The best known bound for quasigroups is SAC^1 due to Wagner [58]. Improvements in this direction would immediately yield improvements in MGS for quasigroups. Furthermore,

a *constructive* membership test would also yield improvements for isomorphism testing of $O(1)$-generated quasigroups. Note that isomorphism testing of $O(1)$-generated groups is known to belong to L [53].

QUESTION 8.3. What is the complexity of Minimum Generating Set for semigroups (in the Cayley table model)? More specifically, is it NP-complete?

References

[1] **Abraham A. Albert**. Quasigroups. I. *Trans. Amer. Math. Soc.* 54(3):507–519, 1943. DOI (9)

[2] **Eric Allender and Ulrich Hertrampf**. Depth reduction for circuits of unbounded fan-in. *Inf. Comput.* 112(2):217–238, 1994. DOI (4)

[3] **Vikraman Arvind and Jacobo Torán**. The complexity of quasigroup isomorphism and the minimum generating set problem. *Algorithms and Computation, 17th International Symposium, ISAAC 2006, Kolkata, India, December 18-20, 2006, Proceedings*, volume 4288 of *Lecture Notes in Computer Science*, pages 233–242. Springer, 2006. DOI (5, 8, 29, 30)

[4] **James Aspnes, Richard Beigel, Merrick Furst, and Steven Rudich**. The expressive power of voting polynomials. *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*, STOC '91, pages 402–409, New Orleans, Louisiana, USA. Association for Computing Machinery, 1991. DOI (13)

[5] **László Babai**. Graph isomorphism in quasipolynomial time [extended abstract]. *STOC'16—Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing*, pages 684–697. ACM, New York, 2016. Preprint of full version at arXiv:1512.03547v2 [cs.DS]. DOI (1)

[6] **László Babai**. On the complexity of canonical labeling of strongly regular graphs. *SIAM J. Comput.* 9(1):212–216, 1980. DOI (27)

[7] **László Babai and Endre Szemerédi**. On the complexity of matrix group problems I. *25th Annual Symp. on Foundations of Computer Science, West Palm Beach, Florida, USA, 24-26 October 1984 (FOCS '84)*, pages 229–240. IEEE Computer Society, 1984. DOI (7, 22, 32)

[8] **László Babai and John Wilmes**. Quasipolynomial-time canonical form for steiner designs. *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 261–270. ACM, 2013. DOI (27)

[9] **David A. Mix Barrington**. Quasipolynomial size circuit classes. *[1992] Proceedings of the Seventh Annual Structure in Complexity Theory Conference*, pages 86–93, 1992. DOI (2, 13)

[10] **David A. Mix Barrington, Peter Kadau, Klaus-Jörn Lange, and Pierre McKenzie**. On the complexity of some problems on groups input as multiplication tables. *J. Comput. Syst. Sci.* 63(2):186–200, 2001. DOI (6, 13, 17–19)

[11] **David A. Mix Barrington and Pierre McKenzie**. Oracle branching programs and Logspace versus P. *Inf. Comput.* 95(1):96–115, 1991. DOI (5, 8, 11)

[12] **David A. Mix Barrington and Howard Straubing**. Complex polynomials and circuit lower bounds for modular counting. *Comput. Complex.* 4:325–338, 1994. DOI (21)

[13] **Paul Beame**. A switching lemma primer. Manuscript, 1994. URL (2)

[14] **Paul Beame**. Lower bounds for recognizing small cliques on CRCW PRAM's. *Disc. Appl. Math.* 29(1):3–20, 1990. DOI (2)

[15] **Raj Chandra Bose**. Strongly regular graphs, partial geometries and partially balanced designs. *Pacific J. Math.* 13(2):389–419, 1963. DOI (10, 27)

[16] **Richard H. Bruck**. Finite nets. II. Uniqueness and imbedding. *Pacific J. Math.* 13(2):421–457, 1963. DOI (10)

[17] **Arkadev Chattopadhyay, Jacobo Torán, and Fabian Wagner**. Graph isomorphism is not AC^0-reducible to group isomorphism. *ACM Trans. Comput. Theory*, 5(4):Art. 13, 13, 2013. Preliminary version appeared in FSTTCS '10. DOI ePrint (2, 3, 5–7, 9, 13, 20, 22, 24, 31)

[18] **Anthony E. Clement, Stephen Majewicz, and Marcos Zyman**. The theory of nilpotent groups. Birkhäuser/Springer, Cham, 2017. DOI (19)

[19] **Nathaniel A. Collins, Joshua A. Grochow, Michael Levet, and Armin Weiß**. Constant depth circuit complexity for generating quasigroups. *Proceedings of the 2024 International Symposium on Symbolic and Algebraic Computation, ISSAC 2024, Raleigh, NC, USA, July 16-19, 2024*, pages 198–207. ACM, 2024. DOI (1)

[20] **Bireswar Das and Dhara Thakkar**. Algorithms for the minimum generating set problem. arXiv:2305.08405, 2023. DOI (8)

[21] **Heiko Dietrich**, **Murray Elder**, **Adam Piggott**, **Youming Qiao**, **and Armin Weiß**. The isomorphism problem for plain groups is in Σ_3^{P}. *39th International Symposium on Theoretical Aspects of Computer Science, STACS 2022, March 15-18, 2022, Marseille, France (Virtual Conference)*, volume 219 of *LIPIcs*, 26:1–26:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. DOI (24)

[22] **Heiko Dietrich and James B. Wilson**. Group isomorphism is nearly-linear time for most orders. *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 457–467, 2022. DOI (6, 7, 22)

[23] **Volkmar Felsch and Joachim Neubüser**. On a programme for the determination of the automorphism group of a finite group. *Computational Problems in Abstract Algebra (Proc. Conf., Oxford, 1967)*, pages 59–60. 1970. DOI (3)

[24] **Flavio Ferrarotti**, **Senén González**, **Klaus-Dieter Schewe**, **and José María Turull-Torres**. Proper hierarchies in polylogarithmic time and absence of complete problems. *Foundations of Information and Knowledge Systems (FoIKS)*, pages 90–105, Cham. Springer International Publishing, 2020. DOI (13)

[25] **Lukas Fleischer**. The Cayley semigroup membership problem. *Theory of Computing*, 18(8):1–18, 2022. DOI (5, 11, 18, 33)

[26] **Joshua A. Grochow and Michael Levet**. On the Parallel Complexity of Group Isomorphism via Weisfeiler-Leman. *Fundamentals of Computation Theory - 24th International Symposium, FCT 2023, Trier, Germany, September 18-21, 2023, Proceedings*, volume 14292 of *Lecture Notes in Computer Science*, pages 234–247. Springer, 2023. Preprint of full version at arXiv:2112.11487 [cs.DS]. DOI (6, 7, 12, 17, 20)

[27] **Joshua A. Grochow and Youming Qiao**. Algorithms for group isomorphism via group extensions and cohomology. *SIAM J. Comput.* 46(4):1153–1216, 2017. Preliminary version in IEEE Conference on Computational Complexity (CCC) 2014. DOI (7)

[28] **Johan Håstad**. Almost optimal lower bounds for small depth circuits. *Proceedings of the 18th Annual ACM Symposium on Theory of Computing (STOC '86), May 28-30, 1986, Berkeley, California, USA*, pages 6–20. ACM, 1986. DOI (2)

[29] **Harald Andrés Helfgott**, **Jitendra Bajpai**, **and Daniele Dona**. Graph isomorphisms in quasi-polynomial time. arXiv:1710.04574, 2017. DOI (1)

[30] **William Hesse**, **Eric Allender**, **and David A. Mix Barrington**. Uniform constant-depth threshold circuits for division and iterated multiplication. *J. Comput. Syst. Sci.* 65(4):695–716, 2002. Special Issue on Complexity 2001. DOI (12)

[31] **D. Holt**, **B. Eick**, **and E. O'Brien**. Handbook of Computational Group Theory. Chapman and Hall/CRC, 2005. DOI (19)

[32] **Jeffrey C. Jackson**, **Adam R. Klivans**, **and Rocco A. Servedio**. Learnability beyond AC^0. *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 776–784, Montreal, Quebec, Canada. Association for Computing Machinery, 2002. DOI (13)

[33] **Mikolás Janota**, **António Morgado**, **and Petr Vojtechovský**. Computing generating sets of minimal size in finite algebras. *J. Symb. Comput.* 119:50–63, 2023. DOI (34)

[34] **Emil Jerábek**. Answer to "is abelian group isomorphism in AC^0?", version: 2020-06-17. Theoretical Computer Science Stack Exchange, June 2020. URL (20)

[35] **A. Donald Keedwell and József Dénes**. Latin squares and their applications. Amsterdam: Elsevier, 2nd ed. Edition, 2015. DOI (9)

[36] **François Le Gall and David J. Rosenbaum**. On the group and color isomorphism problems. arXiv:1609.08253, 2016. DOI (7)

[37] **Michael Levet**. On the complexity of identifying strongly regular graphs. *Australasian J. Combin.* 87:41–67, 2023. URL (5, 6, 9, 25, 27)

[38] **Richard J. Lipton**, **Lawrence Snyder**, **and Yechezkel Zalcstein**. The complexity of word and isomorphism problems for finite groups. Yale University Dept. of Computer Science Research Report # 91, 1977. URL (3, 5, 7)

[39] **Andrea Lucchini and Federico Menegazzo**. Computing a set of generators of minimal cardinality in a solvable group. *J. Symb. Comput.* 17(5):409–420, 1994. DOI (28)

[40] **Andrea Lucchini and Dhara Thakkar**. The minimum generating set problem. *J. Algebra*, 640:117–128, 2024. DOI (4, 8, 28, 29)

[41] **Eugene M. Luks**. Group isomorphism with fixed subnormal chains. arXiv:1511.00151, 2015. DOI (7)

[42] **James F. Lynch**. A depth- size tradeoff for boolean circuits with unbounded fan-in. *Structure in Complexity Theory, Proceedings of the Conference hold at the University of California, Berkeley, California, USA, June 2-5, 1986*, volume 223 of *Lecture Notes in Computer Science*, pages 234–248. Springer, 1986. DOI (2)

[43] **Gary L. Miller**. On the $n^{\log n}$ isomorphism technique (a preliminary report). *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, STOC '78, pages 51–58, San Diego, California, USA. Association for Computing Machinery, 1978. DOI (1, 3, 7, 9, 25, 27, 34)

[44] **Arnold Neumaier**. Strongly regular graphs with smallest eigenvalue $-m$. *Archiv der Mathematik*, 33:392–400, 1979. DOI (27)

[45] **Christos H. Papadimitriou and Mihalis Yannakakis**. On limited nondeterminism and the complexity of the V-C dimension. *J. Comput. Syst. Sci.* 53(2):161–170, 1996. DOI (4, 5, 30, 35)

[46] **Alexander A. Razborov**. An Equivalence between Second Order Bounded Domain Bounded Arithmetic and First Order Bounded Arithmetic, *Arithmetic, proof theory, and computational complexity*. Oxford University Press, May 1993. DOI (2, 31)

[47] **Omer Reingold**. Undirected connectivity in log-space. *J. ACM*, 55(4):17:1–17:24, September 2008. DOI (5, 8, 11)

[48] **Derek J.S. Robinson**. A Course in the Theory of Groups. Springer, 1982. DOI (10)

[49] **David J. Rosenbaum**. Bidirectional collision detection and faster deterministic isomorphism testing. arXiv:1304.3935, 2013. DOI (7)

[50] **Benjamin Rossman**. On the constant-depth complexity of k-clique. *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 721–730, Victoria, British Columbia, Canada. Association for Computing Machinery, 2008. DOI (2)

[51] **Jean-Pierre Serre**. Galois Cohomology. Springer Berlin, Heidelberg, 1st edition, 1997. DOI (29)

[52] **Roman Smolensky**. Algebraic methods in the theory of lower bounds for boolean circuit complexity. *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC), 1987, New York, New York, USA*, pages 77–82. ACM, 1987. DOI (2, 12, 13, 31)

[53] **Bangsheng Tang**. Towards Understanding Satisfiability, Group Isomorphism and Their Connections. PhD thesis, Tsinghua University, 2013. (3, 5, 7, 8, 17, 37)

[54] **Derek R. Taunt**. Remarks on the isomorphism problem in theories of construction of finite groups. *Math. Proc. Cambridge Phil. Soc.* 51(1):16–24, 1955. DOI (23)

[55] **Jacobo Torán**. On the hardness of graph isomorphism. *SIAM J. Comput.* 33(5):1093–1108, 2004. DOI (2)

[56] **Valery Vasil'ev**, **Maria Grechkoseeva**, **and Victor D. Mazurov**. Characterization of the finite simple groups by spectrum and order. *Algebra and Logic*, 48:385–409, December 2009. DOI (18)

[57] **Heribert Vollmer**. Introduction to Circuit Complexity - A Uniform Approach. Texts in Theoretical Computer Science. An EATCS Series. Springer, 1999. DOI (11–13)

[58] **Fabian Wagner**. On the complexity of isomorphism testing for restricted classes of graphs. PhD thesis, Universität Ulm, 2010. DOI (3, 7, 12, 36)

[59] **Marty J. Wolf**. Nondeterministic circuits, space complexity and quasigroups. *Theoret. Comput. Sci.* 125(2):295–313, 1994. DOI (3, 4, 7, 33)

2025:19 TheoretiCS