Electronic Colloquium on Computational Complexity, Report No. 90 (2012)

The Complexity of Intersecting Finite Automata Having Few Final States*

Michael Blondin[1], Andreas Krebs[2], and Pierre McKenzie[1]

[1] Département d'informatique et de recherche opérationnelle, Université de Montréal, Montréal, Québec, Canada
{blondimi, mckenzie}@iro.umontreal.ca
[2] Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Tübingen, Deutschland
krebs@informatik.uni-tuebingen.de

Abstract. The problem of determining whether several finite automata accept a word in common is closely related to the well-studied membership problem in transformation monoids. We raise the issue of limiting the number of final states in the automata intersection problem. For automata with two final states, we show the problem to be ⊕L-complete or NP-complete according to whether a nontrivial monoid other than a direct product of cyclic groups of order 2 is allowed in the automata. We further consider idempotent commutative automata and (abelian, mainly) group automata with one, two or three final states over a singleton or larger alphabet, elucidating the complexity of the intersection nonemptiness and related problems in each case.

1 Introduction

Let $[m]$ denote $\{1, 2, ..., m\}$ and let PS be the *point-spread problem* for transformation monoids, which we define as follows:

Input: $m > 0$, $g_1, \ldots, g_k : [m] \to [m]$ and $S_1, \ldots, S_m \subseteq [m]$.
Question: $\exists g \in \langle g_1, \ldots, g_k\rangle$ such that $i^g \in S_i$ for every $i \in [m]$?

Here $\langle g_1, \ldots, g_k\rangle$ denotes the monoid obtained by closing the set $\{\mathsf{id}_m, g_1, \ldots, g_k\}$ under function composition and i^g denotes the image of i under g.

The PS problem generalizes many problems found in the literature. For example, it generalizes the (transformation monoid) membership problem [Koz77] Memb, the pointset transporter problem [LM88] and the set transporter problem [LM88]. Moreover, it largely amounts to none other than the *finite automata nonemptiness intersection problem*, $\mathsf{AutoInt}$, defined as follows:

Input: finite automata $A_1, \ldots, A_k$ on a common alphabet Σ.
Question: $\exists w \in \Sigma^*$ accepted by A_i for every $i \in [k]$?

* Extended and expanded version of M. Blondin and P. McKenzie, The Complexity of Intersecting Finite Automata Having Few Final States, *Proc. 7th International Computer Science Symposium in Russia*, 2012, to appear.

ISSN 1433-8092

As we note in Proposition 2.1, PS_b, i.e., PS in which some of the S_i can be restricted to have size at most b, has the same complexity as $\mathsf{AutoInt}_b$, i.e., $\mathsf{AutoInt}$ in which the automata have at most b final states, and this holds as well when the monoid in the PS instances and the transition monoids of the automata in the $\mathsf{AutoInt}$ instances are drawn from a fixed monoid variety X. We view PS as mildly more fundamental because it involves a single monoid.

Memb and $\mathsf{AutoInt}$ were shown to be PSPACE-complete by Kozen [Koz77]. Shortly afterwards, the connection with the graph isomorphism problem led to an in-depth investigation of permutation group problems. In particular, Memb was shown to belong to P for groups [FHL80], then to NC^3 for abelian groups [MC87,Mul87], to NC for nilpotent groups [LM88], solvable groups [LM88], groups with bounded non-abelian composition factors [Luk86], and finally all groups [BLS87]. A similar complexity classification of Memb for group-free (or aperiodic) monoids owes to [Koz77,Bea88a,BMT92], who show that Memb for any fixed aperiodic monoid variety is either in AC^0, in P, in NP, or in PSPACE (and complete for that class with very few exceptions).

On the other hand, $\mathsf{AutoInt}$ has received less attention. This is (or might be) due to the fact that $\mathsf{AutoInt}$ is equivalent to Memb when both are intractable, but appears harder than Memb when Memb is efficiently solvable. For example, Beaudry [Bea88b] shows that $\mathsf{AutoInt}$ is NP-complete for abelian groups and for idempotent commutative monoids. Beaudry points out that those two cases are examples where $\mathsf{AutoInt}$ seems strictly harder than Memb (whose complexity is NC^3 for abelian groups and AC^0 for idempotent commutative monoids). Moreover, early results from [Gal76] show that $\mathsf{AutoInt}$ is NP-complete even when Σ is a singleton.

Nevertheless, interesting results concerning $\mathsf{AutoInt}$ are known. For example, the case where k is bounded by a function in the input length was studied in [LR92]. When $k \leq g(n)$, it is proved that the problem is $\mathrm{NSPACE}(g(n)\log n)$-complete under log-space reductions. This arguably provided the first natural complete problems for $\mathrm{NSPACE}(\log^c n)$. Moreover, it was proved by Karakostas, Lipton and Viglas that improving the best algorithms known solving $\mathsf{AutoInt}$ for a constant number of automata would imply $\mathrm{NL} \neq \mathrm{P}$ [KLV03].

More recently, the intersection problem was also studied for regular expressions without binary + operators [Bal02], instead of finite automata. It is shown to be PSPACE-complete for expressions of star height 2 and NP-complete for star height (at most) 1. Finally, the parameterized complexity of a variant of the problem, where Σ^c is considered instead of Σ^*, was examined in [War01]. Different parameterizations of c, k and the size of the automata yield FPT, NP, W[1], W[2] and W[t] complexities. More results on $\mathsf{AutoInt}$ are surveyed in [HK11].

1.1 Our Contribution

We propose PS as the right algebraic formulation of $\mathsf{AutoInt}$. We observe that PS generalizes known problems and we identify PS variants that are both efficiently solvable and interesting. We obtain these variants by restricting the transition monoids of the automata or the number of generators (alphabet size), or by limiting the size of the S_i s (number of final states) to less than 3.

We then mainly investigate monoids that are abelian groups, but we also consider groups, commutative monoids and idempotent monoids. In the case of abelian groups, we need to revisit the equivalences with AGM (abelian permutation group membership) and LCON (feasibility of linear congruences with tiny moduli) [MC87], which have further been investigated recently in the context of log-space counting classes [AV10]. Focussing on the cases involving one or two final states complements Beaudry's hardness proofs for the intersection problem [Bea88b], which require at least three final states. Table 1 summarizes our classification of the complexities of PS(Abelian groups), or equivalently $\mathsf{AutoInt}$ for automata whose transformation monoids are abelian groups.

Table 1. The point-spread and the automata intersection problems for abelian groups.

	Max size of some S_i ; max # of final states		
	1	2	3+
Single generator ; $\lvert\Sigma\rvert = 1$	L-complete	L-complete	NP-complete
Elementary 2-groups	$\oplus$L-complete	$\oplus$L-complete	NP-complete [Bea88b]
Elementary p-groups	Mod_pL-complete	NP-complete	NP-complete [Bea88b]
All abelian groups	NC^3, $\mathrm{FL}^{\mathrm{ModL}}$/poly	NP-complete	NP-complete [Bea88b]

We show that the first line in Table 1 in fact applies as well to nonabelian or nongroup automata over $\Sigma = \{a\}$, and to a class of abelian group automata which we will call *tight abelian group automata*. To the best of our knowledge, Table 1 yields the first efficiently solvable variants of $\mathsf{AutoInt}$. Moreover, it provides characterizations of Mod_pL and thus allows the study of (some) log-space counting classes in terms of automata.

For nonabelian groups and monoids in general, essentially drawing from the literature yields

- $\mathsf{AutoInt}$(Groups) is NP-complete (Proposition 3.2)
- $\mathsf{AutoInt}_1$(Groups) $\in$ NC (Proposition 3.2)
- $\mathsf{AutoInt}_1$(Idempotent and commutative monoids) $\in \mathrm{AC}^0$ (Theorem 4.2).

More strikingly, the two NP-complete entries in the middle column of Table 1 follow from a more general result[3] proved here as Theorem 3.15: if X is any monoid pseudovariety not contained in the 2-elementary abelian groups, then $\mathsf{AutoInt}_2(X)$ is NP-hard. This implies that

- $\mathsf{AutoInt}_2(X)$ is NP-complete for any non-group pseudovariety X, hence
- $\mathsf{AutoInt}_2$(Idempotent and commutative monoids) is NP-complete.

[3] The present paper serves as an extended version of [BM12], with detailed proofs, and it expands on [BM12] by the new Theorem 3.15, largely settling the questions left open in [BM12] concerning automata with two final states.

Finally, we introduce a generalization of $\mathsf{AutoInt}$ by adding $\cup$-clauses. More formally, the problem is to determine whether $\cap_{i=1}^{k} \cup_{j=1}^{k'} \text{Language}(A_{i,j}) \neq \emptyset$. When $k' = 2$ and each automaton possesses one final state, this generalizes the original version of the problem with two final states. In the case of unary languages, we are able to show this variant to be NL-complete and thus suggest this definition to be the right generalization, in-between two and three final states, to avoid complexity blow-ups for some restrictions of $\mathsf{AutoInt}$.

Section 2 presents our notation, defines the relevant problems and relates PS and $\mathsf{AutoInt}$ to some algebraic problems. Section 3 is devoted to the analysis of the complexity of PS and $\mathsf{AutoInt}$ for abelian group automata subject to multiple restrictions. A short Section 4 contains observations about the complexity of PS and $\mathsf{AutoInt}$ in commutative and idempotent monoids. Section 5 concludes and mentions open problems.

2 Preliminaries

2.1 Complexity Theory

We define NC^k (resp. AC^k) as the class of languages accepted by families of bounded (resp. unbounded) fan-in Boolean circuits of polynomial size and depth $O(\log^k n)$. We let $\text{NC} = \cup_k \text{NC}^k$. For NC^k (resp. AC^0), we consider log space (DLOGTIME) uniform circuit families.

A function f is in GapL iff f is log-space many-one reducible to computing the determinant of an integer matrix [AO96]. A language S is in Mod_kL [BDHM92] iff there exists $f \in \#\text{L}$ such that $x \in S \Leftrightarrow f(x) \not\equiv 0 \pmod{k}$. A language S is in ModL [AV10] iff there exists $f \in \text{GapL}, g \in \text{FL}$ such that for all strings x, $g(x) = 0^{p^e}$ for some prime p and $e \in \mathbb{N}$, and $x \in S \Leftrightarrow f(x) \equiv 0 \pmod{|g(x)|}$. For every prime power p^e, $\text{Mod}_{p^e}\text{L} \subseteq \text{ModL} \subseteq \text{NC}^2$, and $\text{FL}^{\text{ModL}} = \text{FL}^{\text{GapL}}$ [AV10].

We use the notation $\leq_m$ (resp. $\leq_T$) for many-one (resp. Turing) reductions. We use $\leq_{\log}$ for log-space reductions, $\leq_{\text{NC}^1}$ for NC^1 reductions and $\leq_{\text{AC}^0}^m$ for AC^0 reductions. Equivalences are defined analogously and denoted by $\equiv$. See [MC87] for more details.

2.2 Basic Definitions and Notation

An *automaton* refers to a deterministic complete finite automaton. Formally, it is a tuple $(\Omega, \Sigma, \delta, \alpha, F)$ where Ω is the set of *states*, Σ is an *alphabet*, $\delta : \Omega \times \Sigma \to \Omega$ is the *transition function*, $\alpha \in \Omega$ is the *initial state* and $F \subseteq \Omega$ is the set of *final states* (*accepting states*). The language of an automaton A is denoted $\text{Language}(A)$. The number of occurrences of σ in a word w is denoted by $|w|_\sigma$. Throughout the paper, the automata defining a problem instance always share an alphabet Σ and we denote its size $|\Sigma|$ by s.

The *transition monoid* $\mathcal{M}(A)$ of an automaton A is the monoid $\langle\{T_\sigma : \sigma \in \Sigma\}\rangle$ where $T_\sigma(\gamma) = \delta(\gamma, \sigma)$. For $w = w_1 \cdots w_\ell$, $T_w = T_{w_\ell} \circ \cdots \circ T_{w_1}$ so for example

$T_{\sigma_1\sigma_2}(\gamma) = T_{\sigma_2}(T_{\sigma_1}(\gamma))$. When $\mathcal{M}(A)$ is a group, and thus a permutation group on Ω, every letter $\sigma \in \Sigma$ has an *order* $\mathrm{ord}(\sigma)$ that may be defined by the order of T_σ in $\mathcal{M}(A)$. However, we prefer considering the automaton A' obtained from removing the states not accessible from the initial state of A. Therefore, we define $\mathrm{ord}(\sigma)$ as the order of T_σ in the transitive permutation group $\mathcal{M}(A')$. For an automaton A, we say that A is an (abelian) group automaton if its transition monoid is an (abelian) group.

An abelian group automaton A will be said to be a *tight abelian group automaton* if $\{v \in \mathbb{Z}_{\mathrm{ord}(\sigma_1)} \times \cdots \times \mathbb{Z}_{\mathrm{ord}(\sigma_s)} : \sigma_1^{v_1} \cdots \sigma_s^{v_s} \in \mathrm{Language}(A)\}$ contains only one element. We note that when $\Sigma = \{a\}$, such automata are directed cycles of size $\mathrm{ord}(a)$, and thus accept only one word of size less than $\mathrm{ord}(a)$. Another family fulfilling this criterion is the set of automata obtained by taking the cartesian product of unary automata working on distinct letters.

Automata are encoded by their transition monoid. We assume any reasonable encoding of monoids, described in terms of their generators, that allows basic operations like composing two transformations and determining the image of a point under a transformation in AC^0.

Let p be a prime. A finite group is a p-group iff its order is a power of p. An abelian group is an abelian elementary p-group iff every non trivial element has order p. A finite group is nilpotent iff it is the direct product of p-groups. [Zas99]

We use lcm for the least common multiple, gcd for the greatest common divisor, n for the input length, and $\mathbb{Z}_q$ for the integers mod q. We say that an integer q is *tiny* if its value is smaller than the input length (i.e. $|q| \leq n$).

2.3 Problems

We define and list the problems mentioned in this paper for ease of reference. Here X is any family of finite monoids, such as all commutative monoids, or all abelian groups, or all groups. In this paper, X will always be a pseudovariety of monoids, i.e., a family of finite monoids closed under finite direct products and under taking homomorphic images of submonoids; see [BMT92] for an argument that such families are a rich and natural choice.

We will not exploit consequences of X being a pseudovariety, except for the following obvious fact: if X is not contained in the least pseudovariety containing the cyclic group C_2, then X either contains a cyclic group C_q for $q > 2$ or an aperiodic monoid (i.e., a monoid that contains no nontrivial group).

$\mathsf{PS}_b(X)$ (Point-spread problem)

Input: $m > 0$, $g_1, \ldots, g_k : [m] \to [m]$ such that $\langle g_1, \ldots, g_k \rangle \in X$, and $S_1, \ldots, S_m \subseteq [m]$, such that $|S_i| \leq b$ or $|S_i| = m$ for every $i \in [m]$.

Question: $\exists g \in \langle g_1, \ldots, g_k \rangle$ such that $i^g \in S_i$ for every $i \in [m]$?

$\mathsf{AutoInt}_b(X)$ (Automata nonemptiness intersection problem)

Input: finite automata $A_1, \ldots, A_k$ on a common alphabet Σ, such that $\mathcal{M}(A_i) \in X$ and A_i has at most b final states for every $i \in [k]$.
Question: $\exists w \in \Sigma^*$ accepted by A_i for every $i \in [k]$?

$\mathsf{Memb}(X)$ (Membership problem)

Input: $m > 0$, $g_1, \ldots, g_k : [m] \to [m]$ such that $\langle g_1, \ldots, g_k \rangle \in X$, and $g : [m] \to [m]$.
Question: $g \in \langle g_1, \ldots, g_k \rangle$?

$\mathsf{PT}(X)$ (Pointset transporter)

Input: $m > 0$, $g_1, \ldots, g_k : [m] \to [m]$ such that $\langle g_1, \ldots, g_k \rangle \in X$, and $b_1, \ldots, b_r \in [m]$ for some $r \leq m$.
Question: $\exists g \in \langle g_1, \ldots, g_k \rangle$ such that $i^g = b_i$ for every $i \in [r]$?

$\mathsf{ST}(X)$ (Set transporter)

Input: $m > 0$, $g_1, \ldots, g_k : [m] \to [m]$ such that $\langle g_1, \ldots, g_k \rangle \in X$, $r \leq m$ and $B \subseteq [m]$.
Question: $\exists g \in \langle g_1, \ldots, g_k \rangle$ such that $\{1^g, 2^g, \ldots, r^g\} \subseteq B$?

LCON (Linear congruences)

Input: $B \in \mathbb{Z}^{k \times l}, b \in \mathbb{Z}^k$, and an integer q presented as a list of its tiny factors $p_1^{e_1}, \ldots, p_r^{e_r}$.
Question: $\exists x \in \mathbb{Z}^l$ satisfying $Bx \equiv b \pmod{q}$?

$\mathsf{LCONNULL}$ (Linear congruences “nullspace”)

Input: $B \in \mathbb{Z}^{k \times l}$, and an integer q presented as a list of its tiny factors $p_1^{e_1}, \ldots, p_r^{e_r}$.

Problem: compute a generating set for the $\mathbb{Z}$-module $\{x \in \mathbb{Z}^l : Bx \equiv 0 \pmod{q}\}$.

$\mathsf{PS}(X)$ and $\mathsf{AutoInt}(X)$ refer to $\mathsf{PS}_b(X)$ and $\mathsf{AutoInt}_b(X)$ with no bound placed on b. Moreover, we refer to b as the number of final states, even in the context of PS. When the modulus q is fixed to a constant, we use the notation LCON_q and $\mathsf{LCONNULL}_q$.

The point-spread problem relates to other problems as follows.

Proposition 2.1. $\mathsf{AutoInt}_b(X) \equiv^m_{\mathrm{AC}^0} \mathsf{PS}_b(X)$ *for any finite monoid variety* X*.*

Proof. $\mathsf{AutoInt}_b(X) \leq^m_{\mathrm{AC}^0} \mathsf{PS}_b(X)$:

Let $A_1, \ldots, A_k$ be the given automata where $A_i = (\Omega_i, \Sigma, \delta_i, \alpha_i, F_i)$ for every $i \in [k]$. Suppose Ω_i and Ω_j are disjoint for every $i \neq j$ and let $\Omega = \Omega_1 \cup \cdots \cup \Omega_k$.

For each $\sigma \in \Sigma$, let g_σ be the transformation action of the letter σ on Ω. For each $\gamma \in \Omega$, let

$$S_\gamma = \begin{cases} F_i & \text{if } \gamma \text{ is the initial state of } A_i, \\ \Omega & \text{if } \gamma \text{ is any other state of } A_i. \end{cases}$$

Let α_i be the initial state of A_i, then there is a word w accepted by every automaton iff ${\alpha_i}^{g_w} \in F_i$ for every $i \in [k]$ iff g_w maps every initial state to a final state. To complete the reduction, one must notice that $|S_\gamma|$ is either equal to $|\Omega|$ or bounded by b. Moreover, $\langle\{g_\sigma : \sigma \in \Sigma\}\rangle \in X$ since it is a submonoid of $\mathcal{M}(A_1) \times \cdots \times \mathcal{M}(A_k)$.

$\mathsf{PS}_b(X) \leq^m_{\mathrm{AC}^0} \mathsf{AutoInt}_b(X)$: For every $i \in [m]$, let $A_i = ([m], \{g_1, \ldots, g_k\}, \delta, i, S_i)$ where $\delta : [m] \times \{g_1, \ldots, g_k\} \to [m]$ maps (j, g_ℓ) to j^{g_ℓ} for every $j \in [m]$, $\ell \in [k]$. When $S_i = [m]$, we do not build any automaton since it would accept Σ^*. We note that there exists $g \in \langle g_1, \ldots, g_k\rangle$ such that $i^g \in S_i$ for every $i \in [m]$ iff g is accepted by every automaton. Moreover, every automaton has at most b final states and $\mathcal{M}(A_i) \in X$. □

Proposition 2.2. $\mathsf{Memb}(X) \leq^m_{\mathrm{AC}^0} \mathsf{PT}(X) \equiv^m_{\mathrm{AC}^0} \mathsf{PS}_1(X)$ *and* $\mathsf{ST}(X) \leq^m_{\mathrm{AC}^0} \mathsf{PS}(X)$.

Proof. We use the same generators for every reduction. For $\mathsf{Memb}(X) \leq^m_{\mathrm{AC}^0} \mathsf{PT}(X)$, we let $b_i = i^g$ for every $i \in [m]$ where g is the given test transformation. For $\mathsf{PT}(X) \leq^m_{\mathrm{AC}^0} \mathsf{PS}_1(X)$, we let $S_i = \{b_i\}$ for every $i \in [r]$ and $S_i = [m]$ otherwise. For $\mathsf{PS}_1(X) \leq^m_{\mathrm{AC}^0} \mathsf{PT}(X)$, if $|S_i| = 1$, we let b_i be the unique element of S_i. To be consistent with the definition, the points should be reordered such that the points transported come first. Finally, for $\mathsf{ST}(X) \leq^m_{\mathrm{AC}^0} \mathsf{PS}(X)$, we let $S_i = B$ for every $i \in [r]$, and $S_i = [m]$ for every i such that $r < i \leq m$. □

Proposition 2.3. *If* $\mathsf{Memb}(X) \in$ NP (PSPACE) *then* $\mathsf{PS}(X) \in$ NP (PSPACE)*.*

Proof. We guess a transformation g such that $i^g \in S_i$ for $i \in [m]$. From there, we run the NP (PSPACE) machine for $\mathsf{Memb}(X)$ to test whether $g \in \langle g_1, \ldots, g_k\rangle$. For the PSPACE result, we use PSPACE = NPSPACE [Sav70]. □

3 Groups and Abelian Groups

We first recall a slick reduction. Let $\mathsf{PointStab}$(Groups) be the problem in which, given the same input as in problem PT(Groups), we must compute a generating set for the pointwise stabilizer of $\{b_1, \ldots, b_r\}$ in $\langle g_1, \ldots, g_k\rangle$, i.e., the subgroup formed of all $h \in \langle g_1, \ldots, g_k\rangle$ such that $b_i^h = b_i$ for $1 \leq i \leq r$.

Proposition 3.1. [Luk90] $\mathsf{PT}(\mathit{Groups}) \leq^T_{\mathrm{AC}^0} \mathsf{PointStab}(\mathit{Groups})$.

Proof. We sketch the proof [Luk90, p. 27] for completeness. Let $g_1, \ldots, g_k$ be permutations of $[m]$ and $b_1, \ldots, b_r \in [m]$. Assuming with no loss of generality that some g_i is the identity permutation, let

$$G = \langle\{(g_s, g_t) : 1 \leq s, t \leq k\}\rangle \cong \langle g_1, \ldots, g_k\rangle \times \langle g_1, \ldots, g_k\rangle$$

act on $[m] \times [m]$ as $(i, j)^{(g_s, g_t)} = (i^{g_s}, j^{g_t})$. Now define x as the permutation that merely flips each pair (i, j), i.e., $(i, j)^x = (j, i)$ for every $(i, j) \in [m] \times [m]$. We prove that the pointwise stabilizer H of $\{(1, b_1), (2, b_2), \ldots, (r, b_r)\}$ in

$$\langle\{(g_s, g_t) : 1 \leq s, t \leq k\} \cup \{x\}\rangle = \langle G \cup \{x\}\rangle$$

is not contained in G iff some $g \in \langle g_1, \ldots, g_k \rangle$ maps i to b_i for $1 \leq i \leq r$.

We first note that any $y \in \langle G \cup \{x\} \rangle$ is of the form

$$y = (g_1, h_1)x(g_2, h_2) \cdots x(g_n, h_n).$$

Moreover, if $y \notin G$ then it must have an odd number of occurences of x since for even number of occurences, we have $(i,j)^y = (i^{g_1 h_2 g_3 \cdots h_n}, j^{h_1 g_2 h_3 \cdots g_n})$ and thus y may be rewritten as an element of G.

$\Rightarrow$) If $H \not\subseteq G$, then there exists $y \in H$ such that $y \notin G$. Moreover $y = (g_1, h_1)x(g_2, h_2) \cdots x(g_n, h_n)$ where x appears an odd number of times. Therefore $yx \in G$ and $(i, b_i)^{yx} = (i, b_i)^x = (b_i, i)$ for $1 \leq i \leq r$.

$\Leftarrow$) Suppose there exists $g \in \langle g_1, \ldots, g_k \rangle$ such that $i^g = b_i$ for $1 \leq i \leq r$. We have $(g, g^{-1})x \in H$ since $(i, b_i)^{(g,g^{-1})x} = (b_i, i)^x = (i, b_i)$. Moreover, $(g, g^{-1})x \notin G$ since the opposite would imply that $x \in G$ which is impossible.

Given this lemma, we compute generators for H by using the pointwise stabilizer oracle gate, and we detect whether H is larger than G by testing whether any generator of H flips a pair $(i,j) \in [m] \times [m]$. $\square$

By the massive work of [BLS87], PointStab(Groups) $\in$ NC. Combined with Propositions 3.1, 2.2 and 2.3, and with the forthcoming Theorem 3.23, this yields:

Proposition 3.2. PS_1(*Groups*) $\in$ NC *and* PS(*Groups*) *are* NP-complete *under* $\leq^m_{\mathrm{AC}^0}$ *reducibility.*

We will see later that PS_2(Groups) is NP-complete. It is shown in [LM88] that PT(Nilpotent groups) $\in$ NC, so that PS_1(Nilpotent groups) $\in$ NC by Proposition 2.2. This implies that both problems belong to NC for abelian groups.

The rest of our investigation of PS in the group case is devoted to abelian groups. We first refine the above NC upper bound for PS_1(Abelian groups) to NC^3, namely the same complexity as Memb(Abelian groups). To achieve this, we give some definitions and lemmata to show that $\mathsf{AutoInt}_1$(Abelian groups) $\leq^T_{\mathrm{NC}^1}$ LCONNULL.

Definition 3.3. *Let* $A = (\Omega, \Sigma, \delta, \alpha, F)$ *be an abelian group automaton. We define* Φ_A *as the following set:*

$$\Phi_A = \left\{ v \in \mathbb{Z}_q^s \,:\, T_{\sigma_1^{v_1} \cdots \sigma_s^{v_s}} \in G_\alpha \right\},$$

where $q = \mathrm{lcm}(\mathrm{ord}(\sigma_1), \ldots, \mathrm{ord}(\sigma_s))$ *and* $\Sigma = \{\sigma_1, \ldots, \sigma_s\}$.

In other words, Φ_A is the set of vectors $(v_1, \ldots, v_s) \in \mathbb{Z}_q^s$ such that reading $\sigma_1^{v_1} \cdots \sigma_s^{v_s}$ from the initial state α leads back to α. Since the language accepted by A is commutative and the order of each letter divides q, the set Φ_A characterizes Language(A).

Lemma 3.4. *Let* $A = (\Omega, \Sigma, \delta, \alpha, F)$ *be an abelian group automaton, then* Φ_A *is a sub* $\mathbb{Z}_q$*-module of* $\mathbb{Z}_q^s$ *where* $q = \mathrm{lcm}(\mathrm{ord}(\sigma_1), \ldots, \mathrm{ord}(\sigma_s))$.

Definition 3.5. *Let $A = (\Omega, \Sigma, \delta, \alpha, F)$ be an abelian group automaton. Let $q = \mathrm{lcm}(\mathrm{ord}(\sigma_1), \ldots, \mathrm{ord}(\sigma_s))$. We define the monoid homomorphism $\phi_A : \Sigma^* \to \mathbb{Z}_q^s$ as:*

$$\phi_A(w) = (|w|_{\sigma_1} \bmod q, \ldots, |w|_{\sigma_s} \bmod q).$$

This homomorphism is alternatively the Parikh image with each component taken modulo q for a well chosen $q \in \mathbb{N}^+$.

Lemma 3.6. *Let $A = (\Omega, \Sigma, \delta, \alpha, F)$ be an abelian group automaton, $\beta \in \Omega$, $0 \leq i \leq s$ and $b_1, \ldots, b_i \in \mathbb{N}$. It is possible to verify whether there exists a word $w \in \Sigma^*$ such that $T_w(\alpha) = \beta$ and $|w_{\sigma_j}| = b_j$ for every $1 \leq j \leq i$ in logarithmic space. Moreover, if such a word exists then it is possible to compute one in logarithmic space.*

Proof. We first note that A may be considered as an undirected graph. Indeed, since $\mathcal{M}(A)$ is a group, traversing an arc labeled by σ in reverse direction is equivalent to applying T_σ^{-1}. Therefore, for every arc (transition) from γ to γ' labeled by σ, we add the arc (γ', γ) labeled by σ^{-1}. Since $\mathcal{M}(A)$ is abelian, we may suppose, without loss of generality, that $\sigma_1, \ldots, \sigma_i$ are read first. Let $\alpha' = T_{w'}(\alpha)$ where $w' = \sigma_1^{b_1} \cdots \sigma_i^{b_i}$ and remove every transition associated to $\sigma_1, \ldots, \sigma_i$. It now suffices to find a path from α' to β in the graph to build a word w such that $T_w(\alpha) = \beta$. Since finding a path in an undirected graph is in FL [Rei05], we can find such word in logarithmic space. ☐

Lemma 3.7. *Let $A = (\Omega, \Sigma, \delta, \alpha, F)$ be an abelian group automaton. A generating set U for Φ_A such that $|U| \leq \mathrm{ord}(\sigma_1) + \ldots + \mathrm{ord}(\sigma_s) + |\Sigma|$ can be computed in logarithmic space.*

Proof. We give the following algorithm:

for $i \leftarrow 1$ to $|\Sigma|$ do
    for $j \leftarrow 0$ to $\mathrm{ord}(\sigma_i) - 1$ do
        compute w (if any) such that $T_w(\alpha) = \alpha$,
            $|w_{\sigma_r}| = 0$ for every $1 \leq r < i$, and $|w_{\sigma_i}| = j$
        output $\phi_A(w)$
    output v such that $v_i = \mathrm{ord}(\sigma_i)$ and $v_r = 0$ for every $r \neq i$

We first note that the algorithm computes at most $\mathrm{ord}(\sigma_1) + \ldots + \mathrm{ord}(\sigma_s) + |\Sigma|$ vectors. Moreover, the word w computed at line 4 is computable in logarithmic space by Lemma 3.6.

We now show that $\langle U \rangle = \Phi_A$. Let $v \in \Phi_A$. We prove by induction on s, that there exists $u_1, \ldots, u_s \in \langle U \rangle$ such that $u_{i,j} = 0$ for every $1 \leq j < i$ and $u_{i,i} = v_i - \sum_{j=1}^{i-1} u_{j,i}$. Before doing so, we notice that the validity of this affirmation implies $v = u_1 + \ldots + u_s$, and thus $v \in \langle U \rangle$.

We observe that there exists $x < q$ such that $v_1 = (v_1 \bmod \mathrm{ord}(\sigma_1)) + x \cdot \mathrm{ord}(\sigma_1)$. Let $u'_1 \in U$ be such that $u'_{1,1} = v_1 \bmod \mathrm{ord}(\sigma_1)$, and let $u_1 = u'_1 + (x \cdot \mathrm{ord}(\sigma_1), 0, \ldots, 0)$. Therefore $u_1 \in \langle U \rangle$ and $u_{1,1} = v_1$. We notice that there exists $v' \in \Phi_A$ such that $v'_1 = v_1 \bmod \mathrm{ord}(\sigma_1)$, since the vector obtained by modifying the first component of v by the value $v_1 \bmod \mathrm{ord}(\sigma_1)$ is in Φ_A. Therefore, line 4 will necessarily generate such a vector u'_1.

Suppose the hypothesis holds for $u_1, \ldots, u_{i-1}$. Let $v' = v - (u_1 + \ldots + u_{i-1})$, then $v' \in \Phi_A$. Moreover $v'_j = 0$ for every $i \leq j < i$ and $v'_i = v_i - \sum_{j=1}^{i-1} u_{j,i}$. Let $u'_i \in U$ be such that $u'_{i,j} = 0$ for every $j < i$ and $u'_{i,i} = v'_i \bmod \mathrm{ord}(\sigma_i)$, then let $u_i = u'_1 + (0, \ldots, y \cdot \mathrm{ord}(\sigma_i), \ldots, 0)$. Let $u_i = u'_1 + y(0, \ldots, \mathrm{ord}(\sigma_i), \ldots, 0)$. Therefore $u_i \in \langle U \rangle$ and $u_{i,i} = v'_i$ for some $y < q$. As stated in the base case, line 4 will generate such a vector u'_i. □

Definition 3.8. *Let V be a submodule of $\mathbb{Z}_q^s$, then*

$$V^\perp = \{u \in \mathbb{Z}_q^s : \forall v \in V \quad v \cdot u = 0\},$$

where $\cdot$ is the usual dot product (i.e. $u \cdot v = (u_1v_1 + \ldots + u_sv_s) \bmod q$).

Proposition 3.9 ([Luo09], see [Blo12] for explicit details). *Let V be a submodule of $\mathbb{Z}_q^s$, then $(V^\perp)^\perp = V$.*

Lemma 3.10. *Let $x, x' \in \mathbb{N}^s$ and let $U = \{u_1, \ldots, u_{|U|}\}$ be a generating set of $\Phi_A^\perp$. Let $q = \mathrm{lcm}(\mathrm{ord}(\sigma_1), \ldots, \mathrm{ord}(\sigma_s))$ and let B be the matrix such that its i^{th} row is u_i. We have*

$$Bx \equiv Bx' \pmod{q} \Leftrightarrow T_w(\alpha) = T_{w'}(\alpha)$$

where $w = \sigma_1^{x_1} \cdots \sigma_s^{x_s}$ and $w' = \sigma_1^{x'_1} \cdots \sigma_s^{x'_s}$.

Proof. $\Rightarrow$) Let $v = \phi_A(w)$ and $v' = \phi_A(w')$, then $B(v - v') \equiv Bv - Bv' \equiv 0 \pmod{q}$ and therefore $v - v' \in \Phi_A^{\perp\perp}$. By Lemma 3.9, we have $v - v' \in \Phi_A$, and therefore $v + \Phi_A = v' + \Phi_A$. Thus, there exists $v'' \in \Phi_A$ such that $v = v' + v''$ and

$$\begin{aligned}
T_w(\alpha) &= T_{\sigma_1^{x_1} \cdots \sigma_s^{x_s}}(\alpha) && \text{(By definition of } w\text{)} \\
&\equiv T_{\sigma_1^{|w|_{\sigma_1} \bmod q} \cdots \sigma_s^{|w|_{\sigma_s} \bmod q}}(\alpha) && (\mathrm{ord}(\sigma_i) \mid q) \\
&= T_{\sigma_1^{v_1} \cdots \sigma_s^{v_s}}(\alpha) && \text{(By definition of } v\text{)} \\
&= T_{\sigma_1^{v'_1 + v''_1 \bmod q} \cdots \sigma_s^{v'_s + v''_s \bmod q}}(\alpha) && (v = v' + v'') \\
&\equiv T_{\sigma_1^{v'_1 + v''_1} \cdots \sigma_s^{v'_s + v''_s}}(\alpha) && (\mathrm{ord}(\sigma_i) \mid q) \\
&\equiv T_{(\sigma_1^{v'_1} \cdots \sigma_s^{v'_s}) \cdot (\sigma_1^{v''_1} \cdots \sigma_s^{v''_s})}(\alpha) && (\mathcal{M}(A) \text{ is abelian}) \\
&\equiv T_{\sigma_1^{v'_1} \cdots \sigma_s^{v'_s}}(\alpha) && (v'' \in \Phi_A) \\
&\equiv T_{w'}(\alpha) && \text{(Symmetric to lines 1–3).}
\end{aligned}$$

We conclude that $T_w(\alpha) = T_{w'}(\alpha)$.

$\Leftarrow$) Since $T_w(\alpha) = T_{w'}(\alpha)$, then $T_w {T_{w'}}^{-1} \in G_\alpha$. Let $u \in \Sigma^*$ be such that $T_u = {T_{w'}}^{-1}$, then $\phi_A(wu) \in \Phi_A$. Since ϕ_A is a homomorphism, we have $\phi_A(w) + \phi_A(u) \in \Phi_A$. By Lemma 3.9 we have $\Phi_A = (\Phi_A)^{\perp\perp}$ and therefore

$$B\phi_A(w) + B\phi_A(u) \equiv B(\phi_A(w) + \phi_A(u)) \equiv 0 \pmod{q},$$

and thus,

$$B\phi_A(w) \equiv B(-\phi_A(u)) \pmod{q}.$$

We conclude that $Bx \equiv Bx' \pmod{q}$ since $x \equiv \phi_A(w) \pmod{q}$ and $x' \equiv \phi_A(w') \equiv -\phi_A(u) \pmod{q}$. $\square$

We may now proceed to a classification of the complexity of $\mathsf{AutoInt}$ for abelian groups.

Theorem 3.11. $\mathsf{AutoInt}_1$(*Abelian groups*) $\leq^T_{\mathrm{NC}^1}$ $\mathsf{LCONNULL}$.

Proof. Let $A_1, \ldots, A_k$ be the given automata and let α_i, β_i be respectively their initial and final states. We build a system of linear congruences for each automaton. We first compute a generating set for Φ_{A_i}. By Lemma 3.7, this can be achieved in logarithmic space. Given this set, we can derive a generating set U_i of $\Phi^\perp_{A_i}$ by calling the oracle for $\mathsf{LCONNULL}$. Let $w_i \in \Sigma^*$ be a word such that $T_{w_i}(\alpha_i) = \beta_i$. By Lemma 3.6, such a word can be computed in logarithmic space. Let B_i be the matrix such that each line is a distinct vector from U_i, and let $b_i = B_i \phi_{A_i}(w_i)$. By Lemma 3.10, $B_i x \equiv b_i \pmod{q_i}$ iff $w = \sigma_1^{x_1} \cdots \sigma_s^{x_s}$ is accepted by automaton A_i where $q_i = \operatorname{lcm}(\operatorname{ord}(\sigma_1), \ldots, \operatorname{ord}(\sigma_s))$. Therefore, there exists a solution $x \in \mathbb{Z}^s$, for every $i \in [k]$, to

$$B_i x \equiv b_i \pmod{q_i} \qquad (*)$$

if and only if a word w is accepted by every automaton. Thus, we reduce the instance of the intersection problem to this instance of LCON:

$$\begin{pmatrix} B_1 & q_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ B_k & 0 & \cdots & q_k \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_s \\ y_1 \\ \vdots \\ y_k \end{pmatrix} \equiv \begin{pmatrix} b_1 \\ \vdots \\ b_k \end{pmatrix} \pmod{\operatorname{lcm}(q_1, \ldots, q_k)}$$

which is equivalent to system $(*)$. We note that $\operatorname{lcm}(q_1, \ldots, q_k)$ can be large, but its factors are tiny since $q_1, \ldots, q_k$ are tiny. Moreover, it is important to note that LCON reduces to $\mathsf{LCONNULL}$ which is hard for NL (and L) [MC87] under $\leq^T_{\mathrm{NC}^1}$ reducibility. Therefore, it is possible to compute this reduction with a NC^1 circuit. Indeed, log-space computations may be carried by calls to the oracle for $\mathsf{LCONNULL}$ and the output may be reduced to an $\mathsf{LCONNULL}$ instance. $\square$

Since $\mathsf{LCONNULL} \in \mathrm{NC}^3$ [MC87] and $\mathsf{LCONNULL} \in \mathrm{FL}^{\mathrm{ModL}}/\mathrm{poly}$ [AV10], we obtain the following corollaries.

Corollary 3.12. $\mathsf{AutoInt}_1$(*Abelian groups*) *is in* NC^3 *and* $\mathrm{FL}^{\mathrm{ModL}}$*/poly.*

By Proposition 2.2, Memb(Abelian groups) $\leq^m_{\mathrm{AC}^0}$ $\mathsf{AutoInt}_1$(Abelian groups). Since Memb(Abelian groups) $\in \mathrm{NC}^3$ [MC87], we obtain a rather tight bound.

We now restrict our abelian groups to elementary abelian p-groups. This allows a characterization of the complexity class $\mathrm{Mod}_p\mathrm{L}$ (denoted $\oplus\mathrm{L}$ when $p = 2$) by the intersection problem, and thus in terms of automata.

Theorem 3.13. $\mathsf{AutoInt}_1$(*Elementary abelian p-groups*) *is* $\mathrm{Mod}_p\mathrm{L}$-complete *under* $\leq^m_{\log}$ *reducibility.*

Proof. Every element of a p-group is either of order 1 or p, therefore we have $\mathrm{lcm}(\mathrm{ord}_i(\sigma_1), \ldots, \mathrm{ord}_i(\sigma_s)) \in \{1, p\}$. Thus, the reduction built in the proof of Theorem 3.11 yields a reduction to $\mathsf{LCONNULL}_p$. Moreover this reduction can be made log-space without any significant modification. Indeed, every computation made in the proof can be achieved in logarithmic space and the NC^1 reduction from LCON to $\mathsf{LCONNULL}$ of [MC87] may be converted to a log-space reduction as noted in [AV10]. Therefore, $\mathsf{AutoInt}_1$(Elementary abelian p-groups) $\leq^T_{\log}$ $\mathsf{LCONNULL}_p$. Since $\mathsf{LCONNULL}_p \in \mathrm{Mod}_p\mathrm{L}$ [BDHM92] and

$$\mathrm{Mod}_p\mathrm{L} = \mathrm{Mod}_p\mathrm{L}^{\mathrm{Mod}_p\mathrm{L}} \ (\mathrm{FMod}_p\mathrm{L} = \mathrm{FL}^{\mathrm{Mod}_p\mathrm{L}}) \ [\mathrm{HRV00}],$$

we obtain $\mathsf{AutoInt}_1$(Elementary abelian p-groups) $\in \mathrm{Mod}_p\mathrm{L}$. Similarly, a log-space reduction from LCON_p is easily obtained by mapping each equation to an automaton. Since LCON_p is complete for $\mathrm{Mod}_p\mathrm{L}$ [BDHM92], it completes the proof. □

We now give the first result of this paper concerning the intersection problem with each automaton having at most two final states. When the transition monoids are restricted to elementary abelian 2-groups, we are able to reduce $\mathsf{AutoInt}_2$ to LCON_2. Therefore, in this case, the problem with two final states per automaton is not harder than with one final state.

Theorem 3.14. $\mathsf{AutoInt}_2$(*Elementary abelian* 2*-groups*) *is* $\oplus\mathrm{L}$-complete *under* $\leq^m_{\log}$ *reducibility.*

Proof. We modify the proof of Proposition 3.11. Let α_i be the initial state and β_i, β'_i the two final states of automaton A_i. We use Proposition 3.11 notation; U_i is a generating set for $\Phi^{\perp}_{A_i}$; $w_i, w'_i \in \Sigma^*$ are words such that $\alpha_i^{w_i} = \beta_i$ and $\alpha_i^{w'_i} = \beta'_i$; B_i is the matrix such that each line is a distinct vector from U_i; $b_i = B_i\phi_{A_i}(w_i)$, and $b'_i = B_i\phi_{A_i}(w'_i)$.

By Lemma 3.10, there exists a solution $x \in \mathbb{Z}^s$ to

$$(B_i x \equiv b_i \pmod 2) \vee (B_i x \equiv b'_i \pmod 2) \quad \forall i \in [k]$$

if and only if a word is accepted by every automaton.

We build this system without the $\vee$-clauses by introducing variables z_i, z_i':

$$\begin{pmatrix} 0 & 1 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 1 \\ B_1 & b_1 & b_1' & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ B_k & 0 & 0 & \cdots & b_k & b_k' \end{pmatrix} \begin{pmatrix} x \\ z_1 \\ z_1' \\ \vdots \\ z_k \\ z_k' \end{pmatrix} \equiv \begin{pmatrix} 1 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \pmod 2 .$$

We note that this system is equivalent to

$$B_i x + z_i b_i + z_i' b_i' \equiv 0 \pmod 2 \quad \forall i \in [k],$$

with constraints $z_i + z_i' \equiv 1 \pmod 2$ for every $i \in [k]$. Since $-z_i b_i \equiv z_i b_i \pmod 2$ and $-z_i' b_i' \equiv z_i' b_i' \pmod 2$, this system is equivalent to

$$B_i x \equiv z_i b_i + z_i' b_i' \pmod 2 \quad \forall i \in [k].$$

Constraints $z_i + z_i' \equiv 1 \pmod 2$ force the selection of either b_i or b_i'. Thus, this system of linear congruences is an instance of LCON_2 which possesses a solution iff there exists a word accepted by every automaton. $\square$

In [BM12], we were only able to resolve the complexity of $\mathsf{AutoInt}_2$ (for general alphabets) in the case of elementary abelian 2-groups. This triggered many open questions concerning $\mathsf{AutoInt}_2$. Here we settle all those questions. In particular, as anticipated, the complexity jumps when we go from $\mathsf{AutoInt}_2$(Elementary abelian 2-groups) to $\mathsf{AutoInt}_2$(Elementary abelian 3-groups). But much to our surprise, the jump is all the way from $\oplus$L-completeness to NP-hardness. And in fact, the jump occurs regardless of how we leave the elementary abelian 2-groups:

Theorem 3.15. *Let X be a monoid pseudovariety not contained in the variety of 2-elementary abelian groups, then $\mathsf{AutoInt}_2(X)$ is hard for* NP *under $\leq^m_{\mathrm{AC}^0}$ reducibility.*

Proof. We have mentioned in Section 2.3 that if X is not contained in the monoid pseudovariety of the 2-elementary abelian groups, then either X contains an aperiodic monoid, or it contains a cyclic group $\mathbb{Z}_p$ for $p > 2$. In both cases here we reduce $\mathsf{CIRCUIT{-}SAT}$ to $\mathsf{AutoInt}(X)$.

Given a circuit, we let Σ be the set of gates of this circuit. In our construction the number of occurrences of the letter σ in a word accepted by all automata will represent the truth value of the gate. We will add automata that check the soundness of the representation, and that check that the output gate according to this representation is assigned the value true. Hence a word will be accepted by all the automata iff there is a valid assignment of truth values to the gates of the circuit that sets the output gate to true.

Suppose that X contains a cyclic group $\mathbb{Z}_p$ for $p > 2$. We assume that the circuit only consists of $\wedge$ and $\neg$ gates. In this case a letter σ should occur in the word 0 or 1 times modulo p, where 0 corresponds to false and 1 to true. For each $\sigma \in \Sigma$ we build an automaton with two final states that verifies whether each letter σ occurs either 0 or 1 times modulo p. Taking the intersection of these automata yields a representation of the valid assignments to the circuit gates.

We build extra automata to validate the computations of the circuit. For each negation gate σ with input gate σ', we build an automaton accepting words w such that $|w|_\sigma + |w|_{\sigma'} \equiv 1 \pmod{p}$. For each $\wedge$ gate with input gates σ' and σ'', we build an automaton accepting words w such that $(|w|_{\sigma'} + |w|_{\sigma''} - 2 \cdot |w|_\sigma) \bmod p \in \{0, 1\}$. In the case $p > 3$ this suffices to check the correct evaluation of the $\wedge$ gate (see Table 2). If $p = 3$, we need to add an extra automaton accepting words w such that $(|w|_{\sigma'} + |w|_{\sigma''} - |w|_\sigma) \bmod p \in \{0, 1\}$ since $1 \equiv -2$. As shown in Table 2, these formulas are satisfied iff the assignment agrees with the $\wedge$ gate.

We build one last automaton accepting words w such that $|w|_\sigma \equiv 1 \pmod{p}$ where σ is the output gate. It remains to notice that the transformation monoid of each automaton is a cyclic group $\mathbb{Z}_p$ and is therefore in X.

Table 2. Formulas for $\wedge$ gates. The middle column shows that when $p > 3$, an automaton $\mathbb{Z}_p$ with accepting states 0 and 1 captures precisely the legal truth value triples that describe the operation of an $\wedge$ gate if the automaton moves one step forward upon reading σ', one step forward upon reading σ'' and two steps backward upon reading σ. When $p = 3$, an automaton corresponding to the rightmost column is required as well, because -2 and $+1$ are not distinguished by the automaton from the middle column.

$\sigma'\sigma''\sigma$	$\sigma' \wedge \sigma'' = \sigma$	$\sigma' + \sigma'' - 2\sigma$ $p > 3$ and $p = 3$	$\sigma' + \sigma'' - \sigma$ p=3
000	**1**	**0**	**0**
001	0	-2	-1
010	**1**	**1**	**1**
011	0	-1	0
100	**1**	**1**	**1**
101	0	-1	0
110	0	2	2
111	**1**	**0**	**1**

Assume X contains an aperiodic monoid. Then V must contain U_1, i.e., the monoid $\{0, 1\}$ under multiplication. This holds because X is closed under taking submonoids. Indeed, consider any nontrivial aperiodic submonoid M then M contains a nontrivial idempotent e, i.e., verifying $e^2 = e \neq 1$. The monoid $\{e, 1\}$ is isomorphic to U_1.

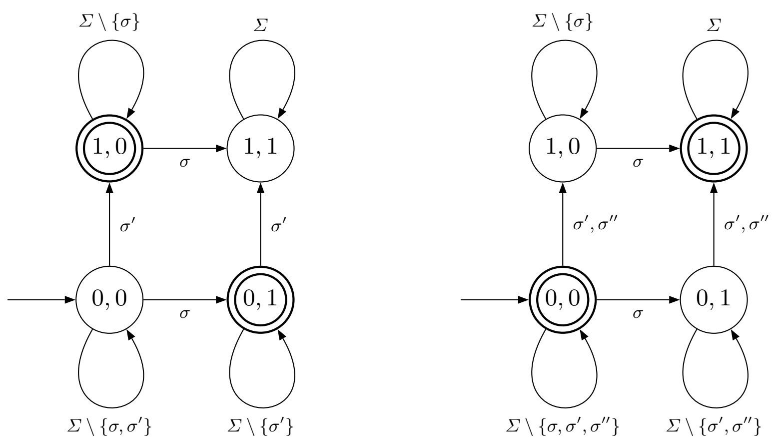
Here we assume that the circuit only consists of $\vee$ and $\neg$ gates. For a word $w \in \Sigma^*$ and a gate $\sigma \in \Sigma$, we consider $|w|_\sigma = 0$ (resp. $|w|_\sigma > 0$) as a 0 (resp. 1) assignment.

For each negation gate σ with input gate σ', we build an automaton accepting words w such that $|w|_{\sigma'} = 0 \Leftrightarrow |w|_\sigma > 0$. For each $\vee$ gate with input gates σ' and σ'', we build an automaton accepting words w such that

$(|w|_{\sigma'} > 0 \vee |w|_{\sigma''} > 0) \Leftrightarrow |w|_\sigma > 0$. These constructions are illustrated in Figure 1.

It remains to build one last automaton accepting words w such that $|w|_\sigma > 0$ where σ is the output gate. The automata built are such that their transition monoid is either U_1 or $U_1 \times U_1$. Since V is closed under finite direct products, this completes the proof.

Fig. 1. Automata for $\neg$ and $\vee$ gates



$\square$

Corollary 3.16. $\mathsf{AutoInt}_2$(*Elementary abelian p-groups*) *for* $p \geq 3$, $\mathsf{AutoInt}_2$(*Abelian groups*), $\mathsf{AutoInt}_2$(*Groups*) *are* NP-complete *under* $\leq^m_{\mathrm{AC}^0}$ *reducibility.*

We may now study the case where Σ consists of a single letter a. Instead of directly considering unary automata, we study the more general case of tight abelian group automata. Before proceeding, we note that the intersection problem over unary languages in general is not harder than for abelian group automata over a unary alphabet

Indeed, an automaton over a singleton alphabet consists of a tail and a cycle. Words accepted by the tail of an automaton may be tested first on the whole collection. If none is accepted, the associated final states are removed and an equivalent cyclic automaton is built.

We first consider a generalization of $\mathsf{AutoInt}$, denoted $\mathsf{AutoInt}(\cup^{k'})$, that consists of determining whether $\cap_{i=1}^{k} \cup_{j=1}^{k'} \mathrm{Language}(A_{i,j}) \neq \emptyset$. We examine the case of $\mathsf{AutoInt}_1(\cup^2)$ that generalizes $\mathsf{AutoInt}_2$, and show it is NL-complete for unary and tight abelian group automata.

We will use the following generalization of the Chinese remainder theorem:

Lemma 3.17. *[Knu81, see p. 277 ex. 3] Let* $a_1, \ldots, a_k \in \mathbb{N}$ *and* $q_1, \ldots, q_k \in \mathbb{N}$. *There exists* $x \in \mathbb{N}$ *such that* $x \equiv a_i \pmod{q_i}$ *for every* $i \in [k]$ *iff* $a_i \equiv a_j \pmod{\gcd(q_i, q_j)}$ *for every* $i, j \in [k]$.

Theorem 3.18. $\mathsf{AutoInt}_1(\bigcup^2$ *Tight abelian group automata*$) \leq^m_{\log} \mathsf{2\text{–}SAT}$.

Proof. Let $A[i,0]$ and $A[i,1]$ be the two automata of the i^{th} $\cup$-clause. Let $v[i,x]$ be the unique vector of $V[i,x] = \{v \in \mathbb{Z}_{\mathrm{ord}_{i,x}(\sigma_1)} \times \cdots \times \mathbb{Z}_{\mathrm{ord}_{i,x}(\sigma_s)} : \sigma_1^{v_1} \cdots \sigma_s^{v_s} \in \mathrm{Language}(A[i,x])\}$ which is computable in log-space. We first note that $A[i,x]$ accepts exactly words $w \in \Sigma^*$ such that $|w|_{\sigma_j} \equiv v[i,x]_j \pmod{\mathrm{ord}_{i,x}(\sigma_j)}$ for every $j \in [s]$, by definition of $V[i,x]$. Therefore, distinct letters are independent and we may find a word accepted by every automaton by verifying restrictions locally on $\sigma_1, \ldots, \sigma_s$. Thus, we have the following equivalences:

$$
\begin{aligned}
&\exists w \text{ such that } w \in \bigcap_{i=1}^{k} \bigcup_{x=0}^{1} \mathrm{Language}(A[i,x]) \\
\Leftrightarrow\ &\exists w\, \exists x \in \{0,1\}^k \text{ such that } w \in \bigcap_{i=1}^{k} \mathrm{Language}(A[i,x_i]) \\
\Leftrightarrow\ &\exists w\, \exists x \in \{0,1\}^k \text{ such that } \bigwedge_{i=1}^{k} \bigwedge_{j=1}^{s} |w|_{\sigma_j} \equiv v[i,x_i]_j \pmod{\mathrm{ord}_{i,x_i}(\sigma_j)} \\
\Leftrightarrow\ &\exists w\, \exists x \in \{0,1\}^k \text{ such that } \bigwedge_{j=1}^{s} \left(\bigwedge_{i=1}^{k} |w|_{\sigma_j} \equiv v[i,x_i]_j \pmod{\mathrm{ord}_{i,x_i}(\sigma_j)} \right) \\
\Leftrightarrow\ &\exists x \in \{0,1\}^k \text{ such that } \bigwedge_{j=1}^{s} \left(\bigwedge_{i=1}^{k} \bigwedge_{i'=1}^{k} C_{i,i',j}(x) \right),
\end{aligned}
$$

where

$$
C_{i,i',j}(x) = \left(v[i,x_i]_j \equiv v[i',x_{i'}]_j \pmod{\gcd(\mathrm{ord}_{i,x_i}(\sigma_j), \mathrm{ord}_{i',x_{i'}}(\sigma_j))} \right).
$$

The last equivalence is a consequence of Lemma 3.17. Therefore, there is a word accepted by every automaton iff this last Boolean expression is satisfiable. For every $i, i' \in [k], j \in [s]$, the truth table of $C_{i,i',j}$ may be computed by evaluating the four congruences. Since $C_{i,i',j}$ depends only on two variables, it is always possible to obtain a 2-CNF. Moreover, the congruences are computable in logarithmic space since the numbers implied are tiny. $\square$

Theorem 3.19. $\mathsf{2\text{–}SAT} \leq^m_{\mathrm{NC}^1} \mathsf{AutoInt}_1(\bigcup^2$ *Abelian groups with* $|\Sigma| = 1)$.

Proof. Let $C(x)$ be the Boolean expression $\bigwedge_{i=1}^{k} C_i(x)$ over $x_1, \ldots, x_m$ where $C_i(x) = (x_{r_i} \oplus b_i) \vee (x_{t_i} \oplus b'_i)$ and $b_i, b'_i \in \{0,1\}$ indicate whether negation must be taken or not.

It is possible to represent an assignment with an integer, assuming it is congruent to 0 or 1 mod the m first primes $p_1, \ldots, p_m$. The remainder of such an integer mod p_i represents the value of the i^{th} variable. Let

$$E_j = \{w \in \{a\}^* \,:\, |w| \equiv 0 \pmod{p_j} \,\vee\, |w| \equiv 1 \pmod{p_j}\}\,,$$
$$X_i = \{w \in \{a\}^* \,:\, |w| \equiv \neg b_i \pmod{p_{r_i}} \,\vee\, |w| \equiv \neg b'_i \pmod{p_{t_i}}\}\,.$$

The language $E_1 \cap \cdots \cap E_m$ represents valid assignments and X_i represents assignments satisfying C_i (but may contain invalid assignments, i.e. not congruent to 0 or 1). The language E_j (resp. X_i) is recognized by the union of two cyclic automata of size p_j (resp. size p_{r_i} and p_{t_i}). It remains to point out that $(E_1 \cap \cdots \cap E_m) \cap (X_1 \cap \cdots \cap X_k) \neq \emptyset$ iff C is satisfiable. □

Corollary 3.20. $\mathsf{AutoInt}_1(\bigcup^2$ *Tight abelian group automata*) *and* $\mathsf{AutoInt}_1(\bigcup^2$ *Abelian groups with* $|\Sigma| = 1$) *are* NL-complete *under* $\leq^m_{\mathrm{NC}^1}$ *reducibility.*

Recall, that 2–$\oplus$SAT is defined similarly to 2–SAT but with $\oplus$ operators instead of $\vee$. It is L-complete under NC^1 reducibility by [MC87] and [JLL76].

Theorem 3.21. $\mathsf{AutoInt}_2$(*Tight abelian group automata*) $\leq^m_{\log}$ 2–$\oplus$SAT.

Proof. We first note that an automaton with two final states may be replaced with the union of two copies of the same automaton, each having one final state. Thus, we may use the proof of Theorem 3.18. However, it remains to show that it is possible to build an expression in 2-$\oplus$CNF (instead of 2-CNF).

To achieve this, we first note that each letter σ_j has the same order in $A[i, 0]$ and $A[i, 1]$ (according to Theorem 3.18 notation). We denote this common order by $\mathrm{ord}_i(\sigma_j)$. Therefore, there is a word accepted by every automaton iff $\bigwedge_{j=1}^{s} \bigwedge_{i=1}^{k} \bigwedge_{i'=1}^{k} C_{i,i',j}(x)$ is satisfiable, where

$$C_{i,i',j}(x) = (v[i, x_i]_j \equiv v[i', x_{i'}]_j \pmod{\gcd(\mathrm{ord}_i(\sigma_j), \mathrm{ord}_{i'}(\sigma_j))})\,.$$

The truth table of $C_{i,i',j}$ may be computed as before by evaluating the four congruences. However, in this case, the modulus is independent of x. Thus, it can be shown that if three of these congruences are true, then all four are. Therefore, $C_{i,i',j}$ can be written solely with the operators $\oplus$ and $\wedge$ as illustrated in Table 3. □

Table 3. Possible expressions for $C_{i,i',j}$

True congruences	Possible expressions
0	0
1	$(x_{i,j} \wedge x_{i',j}), (\neg x_{i,j} \wedge x_{i',j}), (x_{i,j} \wedge \neg x_{i',j}), (\neg x_{i,j} \wedge \neg x_{i',j})$
2	$x_{i,j}, \neg x_{i,j}, x_{i',j}, \neg x_{i',j}, (x_{i,j} \oplus x_{i',j}), (\neg x_{i,j} \oplus x_{i',j})$
4	1

Corollary 3.22. $\mathsf{AutoInt}_2$(*Tight abelian group automata*) *and* $\mathsf{AutoInt}_2$(*Abelian groups with* $|\Sigma| = 1$) *are* L-complete *under* $\leq^m_{\mathrm{NC}^1}$ *reducibility.*

To complete the classification of the intersection problem over unary languages, we argue that it is NP-complete for three final states. A reduction from Monotone 1–in–3 3–SAT [GJ79] may be obtained in a similar fashion to Theorem 3.19. For each clause $(x_1 \vee x_2 \vee x_3)$ we build an automaton with $p_1p_2p_3$ states (and three final states) accepting words $w \in \{a\}^*$ such that

$$(|w| \bmod p_1, |w| \bmod p_2, |w| \bmod p_3) \in \{(1,0,0), (0,1,0), (0,0,1)\}.$$

Theorem 3.23. $\mathsf{AutoInt}_3$(*Tight abelian group automata*) *and* $\mathsf{AutoInt}_3$(*Abelian groups with* $|\Sigma| = 1$) *are* NP-complete *under* $\leq^m_{\mathrm{AC}^0}$ *reducibility.*

4 Some Observations on Commutative and Idempotent Monoids

Here we briefly examine the PS problem for monoids (instead of groups). Recall that a monoid is idempotent iff $x^2 = x$ holds for every element x. We first notice that both PS(Idempotent monoids) and PS(Commutative monoids) are NP-complete. This follows from Propositions 2.2 and 2.3, since their Memb counterparts are NP-complete [Bea88a,Bea88b,BMT92].

Proposition 4.1 ([Bea88a,Bea88b,BMT92]). PS(*Idempotent monoids*) *and* PS(*Commutative monoids*) *are* NP-complete *under* $\leq^m_{\mathrm{AC}^0}$ *reducibility, even for one final state.*

The point-spread problem becomes efficiently solvable when restricted to the variety $\mathbf{J_1}$ of idempotent commutative monoids.

Theorem 4.2. $\mathsf{PS}_1(\mathbf{J_1}) \in \mathrm{AC}^0$.

Proof. We use the technique of [BMT92], for solving Memb($\mathbf{J_1}$), based on the so-called maximal alphabet of a transformation. However, we have to be careful since we are dealing with a partially defined transformation. Let $G = \{g_1, \ldots, g_k\}$ and let b_i be the unique element of S_i. Let $A = \{g \in G : {b_i}^g = b_i \quad \forall i \in [r]\}$ and $a = \prod_{g \in A} g$. Suppose there exists $f \in \langle G \rangle$ such that $i^f = b_i$ for every $i \in [r]$. We first notice that $i^{af} = i^f$ for every $i \in [r]$. Indeed, $i^{af} = i^{fa} = b_i^a = b_i = i^f$. Moreover, we have $h_j \in A$ for any h_j appearing in $f = h_1 \cdots h_l$, since $b_i^{h_j} = i^{fh_j} = i^f = b_i$ for every $i \in [r]$. Thus, $i^{af} = i^{a(h_1 \cdots h_l)} = i^a$ for every $i \in [r]$. Therefore $i^a = i^{af} = i^f = b_i$ for every $i \in [r]$. We conclude that there exists $f \in \langle G \rangle$ such that $i^f = b_i$ for every $i \in [r]$ iff $i^a = b_i$ for every $i \in [r]$. This last test can be carried out easily. ☐

Since $\mathbf{J_1}$ is not contained in the variety of 2-elementary abelian groups, we obtain the following proposition from Theorem 3.15 and Proposition 4.1.

Proposition 4.3. $\mathsf{PS}_2(\mathbf{J_1})$ *and* $\mathsf{PS}_3(\mathbf{J_1})$ *are* NP-complete *under* $\leq^m_{\mathrm{AC}^0}$ *reducibility.*

5 Conclusion and Further Work

This paper raises the issue of limiting the number of accepting states in the automata intersection nonemptiness problem. Limiting that number to fewer than 3 seemed of particular interest because exactly 3 was known to yield NP-completeness in such simple cases as when the automata involved are direct products of cyclic groups of order 2 [Bea88b].

To within the usual hypotheses concerning complexity classes, we completely resolve the complexity of the problem when the number of final states is at most two: the problem is then $\oplus$L-complete or NP-complete, depending on whether no nontrivial monoid other then a direct product of cyclic groups of order 2 occurs. We find interesting, for example, that intersecting two-final-state automata that are direct products of cyclic groups of order 3 is already NP-complete, rather than Mod_3L-complete as we might have expected.

When the number of final states is one, the complexity of the intersection problem naturally bears a close relationship with the complexity of the membership problem in transformation monoids. The membership problem indeed $\leq^m_{\mathrm{AC}^0}$-reduces to the intersection problem (Proposition 2.2) and we show that the case of elementary abelian p-groups is Mod_pL-complete, while the cases of groups and commutative idempotent monoids respectively belong to NC and to AC^0. More generally (Proposition 2.3), any pseudovariety for which membership is NP-complete (resp. PSPACE-complete) has a NP-complete (resp. PSPACE-complete) one-final-state intersection problem. A wealth of such cases are known [BMT92], implying, for example, NP-completeness for aperiodic commutative monoids of threshold two and aperiodic monoids of threshold one, and implying PSPACE-completeness for all aperiodic monoids. We leave open the question of one final state for aperiodic automata whose membership problem lies in the P-complete and NP-hard regions of [BMT92, Fig. 1].

Finally, by restricting the alphabet and relaxing the problem definition, we have identified NL-complete instances of the intersection problem. Here we leave open the questions of the complexity of $\mathsf{PS}_1(\bigcup^2$ Elementary abelian 2-groups) and of $\mathsf{AutoInt}_2$ when $|\Sigma|$ is a constant (e.g. $\Sigma = \{0, 1\}$).

References

AO96. E. Allender and M. Ogihara. Relationships among PL, #L, and the determinant. In *RAIRO - Theoretical Information and Application*, pages 267–278, 1996.

AV10. V. Arvind and T. C. Vijayaraghavan. Classifying problems on linear congruences and abelian permutation groups using logspace counting classes. *Computational Complexlity*, 19:57–98, 2010.

Bal02. S. Bala. Intersection of regular languages and star hierarchy. In *Proc. 29th International Colloquium on Automata, Languages and Programming*, pages 159–169, 2002.

BDHM92. G. Buntrock, C. Damm, U. Hertrampf, and C. Meinel. Structure and importance of logspace-mod class. *Theory of Computing Systems*, 25:223–237, 1992.

Bea88a. M. Beaudry. Membership testing in commutative transformation semigroups. *Information and Computation*, 79(1):84–93, 1988.

Bea88b. M. Beaudry. *Membership testing in transformation monoids*. PhD thesis, McGill University, 1988.

Blo12. M. Blondin. Complexité raffinée du problème d'intersection d'automates. Master's thesis, Université de Montréal, 2012.

BLS87. L. Babai, E. M. Luks, and A. Seress. Permutation groups in NC. In *Proc. 19th annual ACM symposium on Theory of computing*, pages 409–420, 1987.

BM12. M. Blondin and P. McKenzie. The complexity of intersecting finite automata having few final states. In *Proc. 7th International Computer Science Symposium in Russia*, 2012.

BMT92. M. Beaudry, P. McKenzie, and D. Thérien. The membership problem in aperiodic transformation monoids. *J. ACM*, 39(3):599–616, 1992.

FHL80. M. L. Furst, J. E. Hopcroft, and E. M. Luks. Polynomial-time algorithms for permutation groups. In *Proc. 21st Annual Symposium on Foundations of Computer Science*, pages 36–41, 1980.

Gal76. Z. Galil. Hierarchies of complete problems. *Acta Informatica*, 6:77–88, 1976.

GJ79. M.R. Garey and D.S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. W. H. Freeman and Company, 1979.

HK11. M. Holzer and M. Kutrib. Descriptional and computational complexity of finite automata – a survey. *Information and Computation*, 209(3):456–470, 2011.

HRV00. U. Hertrampf, S. Reith, and H. Vollmer. A note on closure properties of logspace mod classes. *Information Processing Letters*, 75:91–93, August 2000.

JLL76. N. D. Jones, Y. E. Lien, and W. T. Laaser. New problems complete for nondeterministic log space. *Theory of Computing Systems*, 10:1–17, 1976.

KLV03. G. Karakostas, R. J. Lipton, and A. Viglas. On the complexity of intersecting finite state automata and NL versus NP. *Theoretical Computer Science*, 302(1-3):257–274, 2003.

Knu81. D.E. Knuth. *The art of computer programming: seminumerical algorithms*, volume 2. Addidon-Wesley, second edition, 1981.

Koz77. D. Kozen. Lower bounds for natural proof systems. In *Proc. 18th Annual Symposium on Foundations of Computer Science*, pages 254–266, 1977.

LM88. E. M. Luks and P. McKenzie. Parallel algorithms for solvable permutation groups. *Journal of Computer and System Sciences*, 37(1):39–62, 1988.

LR92. K.-J. Lange and P. Rossmanith. The emptiness problem for intersections of regular languages. In *Mathematical Foundations of Computer Science*, volume 629 of *Lecture Notes in Computer Science*, pages 346–354. 1992.

Luk86. E. M. Luks. Parallel algorithms for permutation groups and graph isomorphism. In *Proc. 27th Annual Symposium on Foundations of Computer Science*, pages 292–302, 1986.

Luk90. E. M. Luks. *Lectures on polynomial-time computation in groups*. Technical report. College of Computer Science, Northeastern University, 1990. Available at `http://ix.cs.uoregon.edu/~luks/northeasterncourse.pdf`.

Luo09. B. Luong. *Fourier Analysis on Finite Abelian Groups*. Birkhäuser, 2009.

MC87. P. McKenzie and S. A. Cook. The parallel complexity of abelian permutation group problems. *SIAM Journal on Computing*, 16:880–909, 1987.

Mul87. K. Mulmuley. A fast parallel algorithm to compute the rank of a matrix over an arbitrary field. *Combinatorica*, 7:101–104, 1987.

Rei05. O. Reingold. Undirected st-connectivity in log-space. In *Proc. 37th annual ACM symposium on Theory of computing*, pages 376–385, 2005.

Sav70. W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970.

War01. H. T. Wareham. The parameterized complexity of intersection and composition operations on sets of finite-state automata. In *Implementation and Application of Automata*, volume 2088, pages 302–310. 2001.

Zas99. H.J. Zassenhaus. *The Theory of Groups*. Dover Books on Mathematics. Dover Publications, 1999.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de