THE COMPLEXITY OF INTERSECTING FINITE AUTOMATA HAVING FEW FINAL STATES

Michael Blondin, Andreas Krebs, and Pierre McKenzie

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 $\oplus$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 (under the usual hypotheses on complexity classes) the complexity of the intersection nonemptiness and related problems in each case.

Keywords. Finite automata, intersection problem, monoids, logspace, NP-complete, point spread problem.

Subject classification. 68Q15, 68Q25, 68Q17, 03D15, 68Q70.

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:	$m > 0,\ g_1, g_2, \ldots, g_k : [m] \to [m]$ and $S_1, S_2, \ldots, S_m \subseteq [m]$.
Question:	$\exists g \in \langle g_1, g_2, \ldots, g_k \rangle$ such that $i^g \in S_i$ for every $i \in [m]$?

Here $\langle g_1, g_2, \ldots, g_k \rangle$ denotes the monoid obtained by closing the set $\{\mathsf{id}_m, g_1, g_2, \ldots, g_k\}$ 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 (Kozen 1977), Memb, defined as follows:

Input:	$m > 0$, $g_1, g_2, \ldots, g_k, g : [m] \to [m]$.
Question:	$g \in \langle g_1, g_2, \ldots, g_k \rangle$?

As we point out in Section 2.3, it also generalizes the pointset transporter problem (Luks & McKenzie 1988) and the set transporter problem (Luks & McKenzie 1988). Moreover, it largely amounts to none other than the *finite automata nonemptiness intersection problem*, AutoInt, defined as follows:

Input:	finite automata $A_1, A_2, \ldots, A_k$ and a common alphabet Σ.
Question:	$\exists w \in \Sigma^*$ accepted by A_i for every $i \in [k]$?

As we note in Proposition 2.1, PS_b, i.e., PS in which each S_i is restricted to have size m or at most b, has the same complexity as $\mathsf{AutoInt}_b$, i.e., 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 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 AutoInt were shown to be PSPACE-complete by Kozen (Kozen 1977). 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 (Furst *et al.* 1980), then to NC^3 for abelian groups (McKenzie & Cook 1987; Mulmuley 1987), to NC for nilpotent groups (Luks & McKenzie 1988), solvable groups (Luks & McKenzie 1988), groups with bounded non-abelian composition factors (Luks 1986), and finally all groups (Babai *et al.* 1987). A similar complexity classification of Memb for group-free (or aperiodic) monoids owes to (Beaudry 1988a; Beaudry *et al.* 1992; Kozen

1977), 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, AutoInt has received less attention. This is (or might be) due to the fact that AutoInt is equivalent to Memb when both are intractable, but appears harder than Memb when Memb is efficiently solvable. For example, Beaudry (Beaudry 1988b) shows that AutoInt is NP-complete for abelian groups and for idempotent commutative monoids. Beaudry points out that those two cases are examples where 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 (Galil 1976) show that AutoInt is NP-complete even when Σ is a singleton.

Nevertheless, interesting results concerning AutoInt are known. For example, the case where k is bounded by a function in the length of the input to the problem was studied in (Lange & Rossmanith 1992). 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 AutoInt for a constant number k of automata to roughly $o(n^k)$ would imply $\mathrm{NL} \neq \mathrm{P}$ (Karakostas *et al.* 2003).

More recently, the intersection problem was also studied for regular expressions without binary $+$ operators (Bala 2002), 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 (Wareham 2001). 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 AutoInt are surveyed in (Holzer & Kutrib 2011).

1.1. Our Contribution. We propose PS as the right algebraic formulation of 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

Table 1.1: Completeness of the point-spread and the automata intersection problems for abelian groups.

	Max size b of S_i ; max # of final states		
	1	2	3 or more
Single generator; $\lvert\Sigma\rvert = 1$	L	L	NP
Elementary 2-groups	$\oplus$L	$\oplus$L	NP (Beaudry 1988b)
Elementary p-groups	$\mathrm{Mod}_p\mathrm{L}$	NP	NP (Beaudry 1988b)
All abelian groups	$\in \mathrm{NC}^3, \in \mathrm{FL}^{\mathrm{ModL}}/\mathrm{poly}$	NP	NP (Beaudry 1988b)

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) (McKenzie & Cook 1987), which have further been investigated recently in the context of log-space counting classes (Arvind & Vijayaraghavan 2010). Focussing on the cases involving one or two final states complements Beaudry's hardness proofs for the intersection problem (Beaudry 1988b), which require at least three final states. Table 1.1 summarizes our classification of the complexities of PS(Abelian groups), or equivalently AutoInt for automata whose transformation monoids are abelian groups. Table 1.2 summarizes our classification for other pseudovarities of monoids.

In the case of the intersection problem, we show that the first line in Table 1.1 in fact applies as well to 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 1.1 yields the first efficiently solvable variants of AutoInt. Moreover, it provides characterizations of $\mathrm{Mod}_p\mathrm{L}$ 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

Table 1.2: Completeness of the point-spread and the automata intersection problems for monoids.

	Max size b of S_i ; max # of final states		
	1	2	3 or more
Idempotent commutative	$\in$ AC^0	NP	NP (Beaudry 1988b)
Groups	$\in$ NC (Luks 1990)	NP	NP
Commutative	NP (Beaudry *et al.* 1992)	NP (Beaudry *et al.* 1992)	NP (Beaudry *et al.* 1992)
Idempotent	NP (Beaudry *et al.* 1992)	NP (Beaudry *et al.* 1992)	NP (Beaudry *et al.* 1992)
Aperiodic	NP (Beaudry *et al.* 1992)	NP	NP (Beaudry *et al.* 1992)
All monoids	PSPACE (Kozen 1977)	PSPACE (Kozen 1977)	PSPACE (Kozen 1977)

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

More strikingly, the two NP-complete entries in the middle column of Table 1.1 follow from a more general result 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.

Finally, we introduce a generalization of AutoInt by adding $\cup$-clauses. More formally, the problem is to determine whether $\bigcap_{i=1}^{k} \bigcup_{j=1}^{m} \mathrm{L}(A_{i,j}) \neq \emptyset$. When $m = 2$ and each automaton possesses one final state, this generalizes the original version of the problem with two final states. As summarized in Table 1.3, we are able to show this variant to be NL-complete for unary languages, and NP-complete in many other cases.

Section 2 presents our notation, defines the relevant problems and relates PS and AutoInt to some algebraic problems. Section 3 is devoted to the analysis of the complexity of PS and AutoInt

Table 1.3: Completeness of the generalized automata intersection problems for monoids.

	Max # of final states, m of automata per $\cup$-clause	
	1 final state, $m = 2$	1 or more final states, $m \geq 3$
Single generator ; $\lvert\Sigma\rvert = 1$	NL	NP
Elementary abelian p-groups	NP	NP
Groups	NP	NP
Idempotent commutative	NP	NP
Aperiodic	NP (Beaudry *et al.* 1992)	NP (Beaudry *et al.* 1992)
All monoids	PSPACE (Kozen 1977)	PSPACE (Kozen 1977)

for abelian group automata subject to multiple restrictions. A short Section 4 contains observations about the complexity of PS and AutoInt in commutative and idempotent monoids. Section 5 concludes and mentions open problems.

2. Preliminaries

2.1. Complexity Theory. We assume familiarity with L = DSPACE($\log n$) $\subseteq$ NL = NSPACE($\log n$) $\subseteq$ P $\subseteq$ NP $\subseteq$ PSPACE. The class NC^k (resp. AC^k) is the set of languages accepted by families of bounded (resp. unbounded) fan-in Boolean circuits of polynomial size and depth $O(\log^k n)$. Then $\mathrm{NC} = \cup_k \mathrm{NC}^k$. Here the circuit families defining AC^0 and NC^k are taken respectively to be DLOGTIME-uniform (Barrington *et al.* 1990) and logspace-uniform ((Borodin 1977), see (Vollmer 1999) for an extensive treatment of uniformity).

FL is the set of functions computable by deterministic logspace Turing machines. A function $f : \Sigma^* \to \mathbb{N}$ is in #L if there is a logspace nondeterministic Turing machine such that for every input x the number of accepting paths equals $f(x)$. A function $f : \Sigma^* \to \mathbb{Z}$ is in GapL if f is log-space many-one reducible to computing the determinant of an integer matrix (Allender & Ogihara 1996). A language S is in $\mathrm{Mod}_k\mathrm{L}$ (Buntrock *et al.* 1992) if there exists $f \in$ #L such that $x \in S \Leftrightarrow f(x) \not\equiv 0 \pmod{k}$. A language S is in ModL (Arvind & Vijayaraghavan 2010) if there exists $f \in$

GapL, $g \in \mathrm{FL}$ 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, $\mathrm{Mod}_{p^e}\mathrm{L} \subseteq \mathrm{ModL} \subseteq \mathrm{NC}^2$, and $\mathrm{FL}^{\mathrm{ModL}} = \mathrm{FL}^{\mathrm{GapL}}$ (Arvind & Vijayaraghavan 2010).

We use the notation $\leq^m$ (resp. $\leq^T$) for many-one (resp. Turing) reductions. We use $\leq_{\log}$ for log-space reductions, $\leq_{\mathrm{NC}^1}$ for logspace-uniform NC^1 reductions and $\leq_{\mathrm{AC}^0}$ for DLOGTIME-uniform AC^0 reductions. Equivalences are defined analogously and denoted by $\equiv$. In the case of $\leq^T_{\mathrm{NC}^1}$, we follow (McKenzie & Cook 1987) by saying that $A \leq^T_{\mathrm{NC}^1} B$ if by making use of special gates deciding B, A can be decided by a uniform family of circuits in which the n^{th} circuit has depth $O(\log n)$ and has size n (where an oracle gate g is considered to have size equal to one and depth equal to $\log(1 + (\#\text{ of inputs to } g)))$. As noted in (McKenzie & Cook 1987), if $A \leq^T_{\mathrm{NC}^1} B$ and $B \in \mathrm{NC}^k$ then $A \in \mathrm{NC}^k$.

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 $\mathrm{L}(A)$. 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.

A *monoid* is simply a set equipped with an associative binary operation and containing an identity element under that operation. The *transition monoid* $\mathcal{M}(A)$ of an automaton A is the monoid $\langle\{T_\sigma : \sigma \in \Sigma\}\rangle$ formed by closing the set $\{1\} \cup \{T_\sigma : \sigma \in \Sigma\}$ under function composition, where 1 is the identity transformation and $T_\sigma(\gamma) = \delta(\gamma, \sigma)$. For $w = w_1 w_2 \cdots w_\ell$, $T_w = T_{w_\ell} \circ \cdots \circ T_{w_2} \circ T_{w_1}$ so for example $T_{\sigma_1\sigma_2}(\gamma) = T_{\sigma_2}(T_{\sigma_1}(\gamma))$. A *group* is a monoid in which every element g has an inverse g^{-1} such that $g \circ g^{-1} = g^{-1} \circ g = 1$. 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 as the order of T_σ in $\mathcal{M}(A)$, i.e., as the least i such that $T_{\sigma^i} = 1$. However, we prefer considering the automaton A' obtained from removing the states not accessible from the initial state of A. Then $\mathcal{M}(A')$

is transitive on A', and 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

$$\{v \in \mathbb{Z}_{\mathrm{ord}(\sigma_1)} \times \mathbb{Z}_{\mathrm{ord}(\sigma_2)} \times \cdots \times \mathbb{Z}_{\mathrm{ord}(\sigma_s)} \, : \, T_{\sigma_1^{v_1}\sigma_2^{v_2}\cdots\sigma_s^{v_s}}(\alpha) = \beta\}$$

contains only one element for the initial state α and each final state β. We note that when $\Sigma = \{a\}$, such automata are directed cycles of size $\mathrm{ord}(a)$, and thus each final state accepts 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 if its order is a power of p. An abelian group is an abelian elementary p-group if every non trivial element has order p. A finite group is nilpotent if it is the direct product of p-groups (see, for instance, (Zassenhaus 1999)).

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. Finite Monoids and Complexity. We will need very little monoid theory beyond standard linear algebra, but this section is included for completeness and added context.

The universal algebra notion of a *variety* of monoids becomes that of a *pseudovariety* when only finite monoids are studied (Pin 1986): a pseudovariety of finite monoids is any set of finite monoids closed under taking homomorphisms, taking submonoids and forming finite direct products. A pseudovariety of finite monoids is the epitome of a "natural" class: abelian groups, nilpotent groups, solvable groups, all groups, aperiodic monoids (a.k.a. group-free monoids, i.e., those having no nontrivial group as a subset) and any set of monoids whose elements verify a fixed set of identities are pseudovarieties.

In the 1960's and 1970's, pseudovarieties of monoids came into prominence in the study of regular languages, whose combinatorial properties are tied to the algebraic properties of the transition monoid of their minimal automata. Celebrated results include the characterization of star-free regular languages as those having an aperiodic such monoid (Schützenberger 1965). The theory led to a rather complete understanding of regular languages "recognized" by monoids drawn from any pseudovariety that excludes nonsolvable groups (Pin 1986).

In the 1980's, Barrington and Thérien (Barrington 1989; Barrington & Thérien 1988) observed that extending the notion of "recognition" to that of "program-recognition" allows lifting the above theory to the level of NC^1. In the new theory, any pseudovariety that *contains* a nonsolvable group captures NC^1, while the internal structure of NC^1 hinges on the subtle behavior of the remaining pseudovarieties (see (Straubing 1994), leading to Conjecture IX.3.4, whose validity would settle the major open questions about ACC^0 circuits, i.e., AC^0 with MOD_q gates).

Further applications of the theory of finite monoids include proofs of decidability for certain temporal logics (Thérien & Wilke 1998), contributions to our understanding of uniform circuit complexity (Barrington *et al.* 1990; Behle & Lange 2006), links to models of communication complexity (Tesson & Thérien 2005) and, for example, a characterization of the regular languages (with a so-called neutral letter) whose membership can be determined by ACC^0 circuits having a linear number of wires (Koucký *et al.* 2005). For a survey of the issues discussed in the present paragraph, see (Tesson & Thérien 2007).

Problems involving finite monoids were studied for their own sake as well, as evidenced from the many results quoted in our introduction section. In particular, we point out the extent to which the membership problem in varieties of aperiodic monoids captures complexity classes within PSPACE (Beaudry *et al.* 1992; Kozen 1977).

Let C_q for $q \geq 2$ denote the cyclic group with q elements. Let p be prime. The following are well known:

- the elementary abelian p-groups, i.e., the class of finite direct products of C_p, form a pseudovariety,
- the class $\mathbf{J_1}$ of commutative and idempotent monoids, i.e., in which every two elements x and y satisfy $xy = yx$ and $xx = x$, forms a pseudovariety,
- if a pseudovariety X is not contained in the least pseudovariety containing C_2, then X either contains a C_q for $q > 2$ or an aperiodic monoid.

2.4. Problems. We define and list the problems mentioned in this paper for ease of reference. Here X is any pseudovariety of finite monoids.

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

  Input: $m > 0$, $g_1, g_2, \ldots, g_k : [m] \to [m]$ such that $\langle g_1, g_2, \ldots, g_k \rangle \in X$, and $S_1, S_2, \ldots, S_m \subseteq [m]$, such that $|S_i| \leq b$ or $|S_i| = m$ for every $i \in [m]$.

  Question: $\exists g \in \langle g_1, g_2, \ldots, g_k \rangle$ such that $i^g \in S_i$ for every $i \in [m]$?

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

  Input: finite automata $A_1, A_2, \ldots, A_k$ and 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 $i \in [k]$?

- $\mathsf{AutoInt}_b(\bigcup^m X)$ (Generalized automata nonemptiness intersection problem)

Input: finite automata $A_{1,1}, A_{1,2}, \ldots, A_{k,m}$ and a common alphabet Σ, such that $\mathcal{M}(A_{i,j}) \in X$ and $A_{i,j}$ has at most b final states for every $i \in [k], j \in [m]$.
Question: $\exists w \in \Sigma^*$ such that $w \in \bigcap_{i=1}^{k} \bigcup_{j=1}^{m} \mathrm{L}(A_{i,j})$?

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

  Input: $m > 0$, $g_1, g_2, \ldots, g_k : [m] \to [m]$ such that $\langle g_1, g_2, \ldots, g_k \rangle \in X$, and $g : [m] \to [m]$.
  Question: $g \in \langle g_1, g_2, \ldots, g_k \rangle$?

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

  Input: $m > 0$, $g_1, g_2, \ldots, g_k : [m] \to [m]$ such that $\langle g_1, g_2, \ldots, g_k \rangle \in X$, $\{\iota_1, \iota_2, \ldots, \iota_r\} \subseteq [m]$, and $b_1, b_2, \ldots, b_r \in [m]$ for some $r \leq m$.
  Question: $\exists g \in \langle g_1, g_2, \ldots, g_k \rangle$ such that ${\iota_i}^g = b_i$ for every $i \in [r]$?

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

  Input: $m > 0$, $g_1, g_2, \ldots, g_k : [m] \to [m]$ such that $\langle g_1, g_2, \ldots, g_k \rangle \in X$, $r \leq m$ and $B \subseteq [m]$.
  Question: $\exists g \in \langle g_1, g_2, \ldots, g_k \rangle$ 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 by its factorization $p_1^{e_1}, p_2^{e_2}, \ldots, p_r^{e_r}$ such that for every $i \in [r]$ the integers p_i and e_i are tiny.
  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 by its factorization $p_1^{e_1}, p_2^{e_2}, \ldots, p_r^{e_r}$ such that for every $i \in [r]$ the integers p_i and e_i are tiny.

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 and the automata intersection problem relate 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 $A_1, A_2, \ldots, A_k$ 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 $\Omega = \Omega_1 \cup \Omega_2 \cup \cdots \cup \Omega_k$.

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 $\mathcal{M}(A_1) \times \mathcal{M}(A_2) \times \cdots \times \mathcal{M}(A_k)$.

$\mathsf{PS}_b(X) \leq^m_{\mathrm{AC}^0} \mathsf{AutoInt}_b(X)$: For every $i \in [m]$, let $A_i = ([m], \{g_1, g_2, \ldots, g_k\}, \delta, i, S_i)$ where $\delta : [m] \times \{g_1, g_2, \ldots, g_k\} \to [m]$ 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^*. If no automata are built, then build a trivial automaton accepting Σ^*. We note that there exists $g \in \langle g_1, g_2, \ldots, g_k \rangle$ 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 $\iota_i = i$ and $b_i = i^g$ 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 $i \in \{\iota_1, \iota_2, \ldots, \iota_r\}$ and $S_i = [m]$ otherwise. For $\mathsf{PS}_1(X) \leq^m_{\mathrm{AC}^0} \mathsf{PT}(X)$, if $|S_i| = 1$, we let $\iota_i = i$ and b_i be the unique element of S_i. 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 $g \in \langle g_1, g_2, \ldots, g_k \rangle$. For the PSPACE result, we use PSPACE = NPSPACE (Savitch 1970). □

PROPOSITION 2.4. $\mathsf{AutoInt}_b(X) \leq^m_{\mathrm{AC}^0} \mathsf{AutoInt}_{\lceil b/m \rceil}(\bigcup^m X)$ *for every* $b \geq m > 1$.

PROOF. Let $A_1, A_2, \ldots, A_k$ be the given automata where $A_i = (\Omega_i, \Sigma, \delta_i, \alpha_i, F_i)$ for every $i \in [k]$. For every $i \in [k]$, we build $A_{i,1}, A_{i,2}, \ldots, A_{i,m}$, m copies of A_i. The first $m-1$ copies each keep $\lceil b/m \rceil$ distinct final states from A_i, and the last automaton keeps the remaining $b - (m-1)\lceil b/m \rceil$ final states. Clearly the union of these m copies accept the language of the original automaton. Moreover, $A_{i,j}$ has at most $\lceil b/m \rceil$ final states and $\mathcal{M}(A_{i,j}) = \mathcal{M}(A_i)$ as the transition monoid is independent of the final states. □

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 $\{b_1, b_2, \ldots, b_r\}$ in $\langle g_1, g_2, \ldots, g_k\rangle$, i.e., the subgroup formed of all $h \in \langle g_1, g_2, \ldots, g_k\rangle$ such that $b_i^h = b_i$ for $1 \le i \le r$.

Proposition 3.1 (Luks 1990). *PT(Groups) $\le^T_{\mathrm{AC}^0}$ $\mathsf{PointStab}$(Groups).*

Proof. We sketch the proof (Luks 1990, p. 27) for completeness. Let $g_1, g_2, \ldots, g_k$ be permutations of $[m]$ and $b_1, b_2, \ldots, b_r \in [m]$. Assuming with no loss of generality that some g_i is the identity permutation, let

$$G = \langle \{(g_s, g_t) : 1 \le s, t \le k\}\rangle \cong \langle g_1, g_2, \ldots, g_k\rangle \times \langle g_1, g_2, \ldots, g_k\rangle$$

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 claim the following:

Claim: The pointwise stabilizer H of $\{(1,b_1), (2,b_2), \ldots, (r,b_r)\}$ in

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

is not contained in G iff some $g \in \langle g_1, g_2, \ldots, g_k\rangle$ maps i to b_i for $1 \le i \le r$.

To prove the claim, we first note that any $y \in \langle G \cup \{x\}\rangle$ can be expressed as

$$y = (f_1, h_1)x(f_2, h_2)\cdots x(f_n, h_n)$$

for $f_i, h_i \in \langle g_1, g_2, \ldots, g_k\rangle$. Moreover, if $y \notin G$ then it must have an odd number of occurrences of x since for an even number of occurrences, we have $(i,j)^y = (i^{f_1 h_2 f_3 \cdots h_n}, j^{h_1 f_2 h_3 \cdots f_n})$ 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 $y = (f_1, h_1)x(f_2, h_2)\cdots x(f_n, h_n)$ where $f_i, h_i \in \langle g_1, g_2, \ldots, g_k\rangle$

and 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 $g \in \langle g_1, g_2, \ldots, g_k \rangle$ 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 the claim, 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 work of (Babai *et al.* 1987), which appeals to the massive classification of finite simple groups, $\mathsf{PointStab}$(Groups) $\in$ NC. Combined with Proposition 3.1, Proposition 2.2 and Proposition 2.3, and with the forthcoming Theorem 3.25, this yields:

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

We will see later that PS_2(Groups) is NP-complete. It is shown in (Luks & McKenzie 1988) 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) $\leq^T_{\mathrm{NC}^1}$ $\mathsf{LCONNULL}$.

DEFINITION 3.3. *Let* $A = (\Omega, \Sigma, \delta, \alpha, F)$ *be an abelian group automaton. We define* $G_\alpha = \{T_w : w \in \Sigma^* \wedge T_w(\alpha) = \alpha\}$ *the stabilizer of* α, *and* Φ_A *as the following set:*

$$\Phi_A = \left\{ v \in \mathbb{Z}_q^s \, : \, T_{\sigma_1^{v_1}\sigma_2^{v_2}\cdots\sigma_s^{v_s}} \in G_\alpha \right\},$$

where $q = \operatorname{lcm}(\operatorname{ord}(\sigma_1), \operatorname{ord}(\sigma_2), \ldots, \operatorname{ord}(\sigma_s))$ *and* $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_s\}$.

In other words, Φ_A is the set of vectors $(v_1, v_2, \ldots, v_s) \in \mathbb{Z}_q^s$ such that reading $\sigma_1^{v_1}\sigma_2^{v_2}\cdots\sigma_s^{v_s}$ 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 $\mathrm{L}(A)$. The following is clear.

PROPOSITION 3.4. *Let* $A = (\Omega, \{\sigma_1, \sigma_2, \ldots, \sigma_s\}, \delta, \alpha, F)$ *be an abelian group automaton, then* Φ_A *is a sub* $\mathbb{Z}_q$*-module of* $\mathbb{Z}_q^s$ *where* $q = \mathrm{lcm}(\mathrm{ord}(\sigma_1), \mathrm{ord}(\sigma_2), \ldots, \mathrm{ord}(\sigma_s))$.

DEFINITION 3.5. *Let* $A = (\Omega, \{\sigma_1, \sigma_2, \ldots, \sigma_s\}, \delta, \alpha, F)$ *be an abelian group automaton. Let* $q = \mathrm{lcm}(\mathrm{ord}(\sigma_1), \mathrm{ord}(\sigma_2) \ldots, \mathrm{ord}(\sigma_s))$. *We define the monoid homomorphism* $\phi_A : \Sigma^* \to \mathbb{Z}_q^s$ *as:*

$$\phi_A(w) = (|w|_{\sigma_1} \bmod q, |w|_{\sigma_2} \bmod q, \ldots, |w|_{\sigma_s} \bmod q).$$

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

LEMMA 3.6. *Let* $A = (\Omega, \{\sigma_1, \sigma_2, \ldots, \sigma_s\}, \delta, \alpha, F)$ *be an abelian group automaton,* $\beta \in \Omega$, $0 \leq i \leq s$ *and* $b_1, b_2, \ldots, b_i \in \mathbb{N}$. *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 $\sigma_1, \sigma_2, \ldots, \sigma_i$ are read first. Let $\alpha' = T_{w'}(\alpha)$ where $w' = \sigma_1^{b_1}\sigma_2^{b_2}\cdots\sigma_i^{b_i}$. Remove every transition associated to $\sigma_1, \sigma_2, \ldots, \sigma_i$ to make sure these letters are not used again so that we may obtain $|w|_{\sigma_j} = b_j$ for every $1 \leq j \leq i$. 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 (Reingold 2005), we can find such a word in logarithmic space. □

LEMMA 3.7. *Let* $A = (\Omega, \{\sigma_1, \sigma_2, \ldots, \sigma_s\}, \delta, \alpha, F)$ *be an abelian group automaton. A generating set* U *for* Φ_A *such that* $|U| \leq \mathrm{ord}(\sigma_1)+\mathrm{ord}(\sigma_2)+\ldots+\mathrm{ord}(\sigma_s)+s$ *can be computed in logarithmic space.*

PROOF. We give the following algorithm:

1. for $i \leftarrow 1$ to $|\Sigma|$ do
2.   for $j \leftarrow 0$ to $\mathrm{ord}(\sigma_i) - 1$ do
3.   compute w (if any) such that $T_w(\alpha) = \alpha$,
   $|w|_{\sigma_r} = 0$ for every $1 \leq r < i$, and $|w|_{\sigma_i} = j$
4.   output $\phi_A(w)$
5.   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 a set U having at most $\mathrm{ord}(\sigma_1) + \mathrm{ord}(\sigma_2) + \ldots + \mathrm{ord}(\sigma_s) + |\Sigma|$ vectors and such that $U \subseteq \Phi_A$ by definition. Moreover, the word w computed at line 3 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 $u_1, u_2, \ldots, u_s \in \langle U \rangle$ 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 note that this statement implies $v = u_1 + u_2 + \ldots + u_s$, and thus $v \in \langle U \rangle$.

We observe that there exists $0 \leq x < q = \mathrm{lcm}(\mathrm{ord}(\sigma_1), \mathrm{ord}(\sigma_2), \ldots, \mathrm{ord}(\sigma_s))$ 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 $u_1 = u_1' + (x \cdot \mathrm{ord}(\sigma_1), 0, \ldots, 0)$. Then $u_1 \in \langle U \rangle$ 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 $u_1, u_2, \ldots, u_{i-1}$. Let $v' = v - (u_1 + u_2 + \ldots + u_{i-1})$, 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 $u'_{i,i} = v'_i \bmod \operatorname{ord}(\sigma_i)$. Let $u_i = u'_1 + (0, \ldots, y \cdot \operatorname{ord}(\sigma_i), \ldots, 0)$. 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. $u \cdot v = (u_1 v_1 + u_2 v_2 + \ldots + u_s v_s) \bmod q$).

We need the following result in the next lemma proof. It can be obtained with basic character theory of finite abelian groups and Pontryagin duality. It is known under different names and notation in the mathematics literature, therefore we prove it for completeness.

PROPOSITION 3.9. *Let V be a submodule of $\mathbb{Z}_q^s$, then $(V^{\perp})^{\perp} = V$.*

PROOF. Let G be a finite abelian group. A character of G is a homomorphism from G to the multiplicative group $\mathbb{C}^{\times}$. Let $\hat{G}$ be the group of characters of G. It is well known that $G \cong \hat{G}$ since G is a finite abelian group (Luong 2009, p. 52 Corollary 3.1.2). Let $\hat{\hat{G}}$ be the group of characters of the finite abelian group $\hat{G}$. Even though we know that $G \cong \hat{\hat{G}}$ by the previous fact, we may define a canonical isomorphism betwen G and $\hat{\hat{G}}$ (as opposed to the case $G \cong \hat{G}$). Let $\kappa : G \to \hat{\hat{G}}$ be defined by $\kappa(g) = \kappa_g$ where $\kappa_g(\chi) = \chi(g)$. Then κ is the so-called natural isomorphism between G and $\hat{\hat{G}}$ (Luong 2009, p. 54 ex. 12).

Let H be a subgroup of G. Let $H^{\#} = \{\chi \in \hat{G} : \chi(H) = 1\}$ and $(H^{\#})^{\#} = \{\psi \in \hat{\hat{G}} : \chi(H^{\#}) = 1\}$. We show the known fact that $H \cong (H^{\#})^{\#}$ (given as an exercice in (Conrad 2013, p. 12 ex. 13) for example). Let $h \in H$ and $\chi \in H^{\#}$, then $\kappa(h)(\chi) = \kappa_h(\chi) = \chi(h) = 1$ by definition of χ. Therefore $\kappa_h \in (H^{\#})^{\#}$ and κ induces an injective homomorphism from H to $(H^\#)^\#$. It remains to show that $|H| = |(H^\#)^\#|$. We have

$$\begin{array}{rcll}
|(H^\#)^\#| & = & |\hat{\hat{G}}|/|\widehat{H^\#}| & (\text{by } \widehat{H^\#} \cong \hat{\hat{G}}/(H^\#)^\#) \\
& = & |G|/|H^\#| & (\text{by } G \cong \hat{\hat{G}} \text{ and } H^\# \cong \widehat{H^\#}) \\
& = & |G|/|\widehat{G/H}| & (\text{by } H^\# \cong \widehat{G/H}\ ) \\
& = & |G|/(|G|/|H|) & (\text{by } G/H \cong \widehat{G/H}) \\
& = & |H|. &
\end{array}$$

The identities used in the first and third equalities can be found in (Luong 2009, p. 54 ex. 10) for example.

Let $\omega_q = e^{(2\pi i)/q}$ and $\chi_u(v) = \omega_q^{u \cdot v}$ for all $u, v \in \mathbb{Z}_q^s$. We have $\widehat{\mathbb{Z}_q^s} = \{\chi_u : u \in \mathbb{Z}_q^s\}$ (Luong 2009, p. 53) and $\mathbb{Z}_q^s \cong \widehat{\mathbb{Z}_q^s}$ with $u \mapsto \chi_u$. Moreover $\chi_u(v) = 1$ iff $u \cdot v = 0$, and therefore $\chi_u \in V^\#$ iff $u \in V^\perp$. Let $u \in \mathbb{Z}_q^s$, then

$$\begin{array}{rcl}
u \in (V^\perp)^\perp & \Leftrightarrow & \forall v \in V^\perp\ v \cdot u = 0 \\
& \Leftrightarrow & \forall v \in V^\perp\ \chi_v(u) = 1 \\
& \Leftrightarrow & \forall \chi_v \in V^\#\ \chi_v(u) = 1 \\
& \Leftrightarrow & \kappa_u(V^\#) = 1 \\
& \Leftrightarrow & \kappa(u) \in (V^\#)^\#.
\end{array}$$

Since κ is a bijection, we have $\kappa((V^\perp)^\perp) = (V^\#)^\#$. Therefore κ induces an isomorphism from $(V^\perp)^\perp$ to $(V^\#)^\#$, and $(V^\perp)^\perp \cong (V^\#)^\# \cong V$. Let $v \in V$ and $v' \in V^\perp$ then $v' \cdot v = v \cdot v' = 0$. Thus, $v \in (V^\perp)^\perp$ and $V \subseteq (V^\perp)^\perp$. Since $(V^\perp)^\perp \cong V$, we conclude that $(V^\perp)^\perp = V$. □

LEMMA 3.10. *Let $x, x' \in \mathbb{N}^s$ and let $U = \{u_1, u_2, \ldots, u_{|U|}\}$ be a generating set of $\Phi_A^\perp$. Let $q = \operatorname{lcm}(\operatorname{ord}(\sigma_1), \operatorname{ord}(\sigma_2), \ldots, \operatorname{ord}(\sigma_s))$ 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 $w = \sigma_1^{x_1}\sigma_2^{x_2} \cdots \sigma_s^{x_s}$ and $w' = \sigma_1^{x'_1}\sigma_2^{x'_2} \cdots \sigma_s^{x'_s}$.

PROOF. Suppose that $Bx \equiv Bx'$. 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 Proposition 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

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

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

We show the opposite direction. Suppose $T_w(\alpha) = T_{w'}(\alpha)$, 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 Proposition 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 $x' \equiv \phi_A(w') \equiv -\phi_A(u) \pmod{q}$. □

We may now proceed to a classification of the complexity of AutoInt for abelian groups.

THEOREM 3.11. $\mathsf{AutoInt}_1$(*Abelian groups*) $\leq$ LCONNULL *for* $\leq \in \{\leq^T_{\mathrm{NC}^1}, \leq^T_{\log}\}$.

PROOF. We first note that LCON reduces to LCONNULL which is hard for NL (and L) (McKenzie & Cook 1987) under $\leq^T_{\mathrm{NC}^1}$ reducibility. Moreover these reductions may be converted to log-space reductions as noted in (Arvind & Vijayaraghavan 2010). Therefore, log-space and LCON computations may be converted to instances of $\mathsf{LCONNULL}$ and computed with oracle gates for $\mathsf{LCONNULL}$.

Let $A_1, A_2, \ldots, A_k$ 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 $\Phi_{A_i}^{\perp}$ 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 $w = \sigma_1^{x_1}\sigma_2^{x_2}\cdots\sigma_s^{x_s}$ is accepted by automaton A_i where $q_i = \mathrm{lcm}(\mathrm{ord}(\sigma_1), \mathrm{ord}(\sigma_2), \ldots, \mathrm{ord}(\sigma_s))$. 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:

$$\begin{pmatrix} B_1 & q_1 & 0 & \cdots & 0 \\ B_2 & 0 & q_2 & \cdots & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ B_k & 0 & 0 & \cdots & q_k \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_s \\ y_1 \\ y_2 \\ \vdots \\ y_k \end{pmatrix} \equiv \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \end{pmatrix} \pmod{\mathrm{lcm}(q_1, q_2, \ldots, q_k)}$$

which is equivalent to system $(*)$. We note that $\mathrm{lcm}(q_1, q_2, \ldots, q_k)$ can be large, but its factors are tiny since $q_1, q_2, \ldots, q_k$ are tiny. □

Since $\mathsf{LCONNULL} \in \mathrm{NC}^3$ (McKenzie & Cook 1987) and $\mathsf{LCONNULL} \in \mathrm{FL}^{\mathrm{ModL}}/\mathrm{poly}$ (Arvind & Vijayaraghavan 2010), we obtain the following corollaries.

Corollary 3.12. $\mathsf{AutoInt}_1$(*Abelian groups*) *is in both* 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) $\in \mathrm{NC}^3$ (McKenzie & Cook 1987), 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 $\mathrm{lcm}(\mathrm{ord}_i(\sigma_1), \mathrm{ord}_i(\sigma_2), \ldots, \mathrm{ord}_i(\sigma_s)) \in \{1, p\}$. Thus, the reduction built in the proof of Theorem 3.11 yields a reduction to $\mathsf{LCONNULL}_p$. Therefore, $\mathsf{AutoInt}_1$(Elementary abelian p-groups) $\leq^T_{\log}$ $\mathsf{LCONNULL}_p$. Since $\mathsf{LCONNULL}_p \in \mathrm{Mod}_p\mathrm{L}$ (Buntrock *et al.* 1992) and $\mathrm{Mod}_p\mathrm{L} = \mathrm{Mod}_p\mathrm{L}^{\mathrm{Mod}_p\mathrm{L}}$ ($\mathrm{FMod}_p\mathrm{L} = \mathrm{FL}^{\mathrm{Mod}_p\mathrm{L}}$) (Hertrampf *et al.* 2000), we obtain $\mathsf{AutoInt}_1$(Elementary abelian p-groups) $\in \mathrm{Mod}_p\mathrm{L}$. Similarly, a many-one log-space reduction from LCON_p is easily obtained by mapping each equation to an automaton. Since LCON_p is complete for $\mathrm{Mod}_p\mathrm{L}$ (Buntrock *et al.* 1992), 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 Theorem 3.11. Let α_i be the initial state and β_i, β'_i the two final states of automaton A_i. We use Theorem 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:

$$\begin{pmatrix} 0 & 1 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 1 \\ B_1 & b_1 & b'_1 & \cdots & 0 & 0 \\ B_2 & b_2 & b'_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ B_k & 0 & 0 & \cdots & b_k & b'_k \end{pmatrix} \begin{pmatrix} x \\ z_1 \\ z'_1 \\ z_2 \\ z'_2 \\ \vdots \\ z_k \\ z'_k \end{pmatrix} \equiv \begin{pmatrix} 1 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \pmod 2 .$$

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. Hardness follows from Theorem 3.13. □

In a preliminary version of the present work (Blondin & McKenzie 2012), 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 elementary abelian 2-groups, then* $\mathsf{AutoInt}_2(X)$ *is hard for* NP *under* $\leq^m_{\mathrm{AC}^0}$ *reducibility.*

PROOF. We have mentioned in Section 2.4 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 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.

Contains a cyclic group: 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 3.1). If $p = 3$, we need to add an extra automaton accepting every words w such that $(|w|_{\sigma'} + |w|_{\sigma''} - |w|_\sigma) \bmod p \in \{0, 1\}$ since $1 \equiv -2$. As shown in Table 3.1, these formulas are satisfied iff the assignment agrees with the $\wedge$ gate.

We build one last automaton accepting words w such that

Table 3.1: Formulas for $\wedge$ gates when $\mathbb{Z}_p \in X$. 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.

$\sigma'\sigma''\sigma$	validity of		
	$\sigma' \wedge \sigma'' = \sigma$	$\sigma' + \sigma'' - 2\sigma \equiv 0, 1$ $p > 3$ and $p = 3$	$\sigma' + \sigma'' - \sigma \equiv 0, 1$ $p = 3$
000	✔	✔	✔
001	✗	✗ (if $p > 3$) / ✓ ($p = 3$)	✗
010	✔	✔	✔
011	✗	✗	✓
100	✔	✔	✔
101	✗	✗	✓
110	✗	✗	✗
111	✔	✔	✔

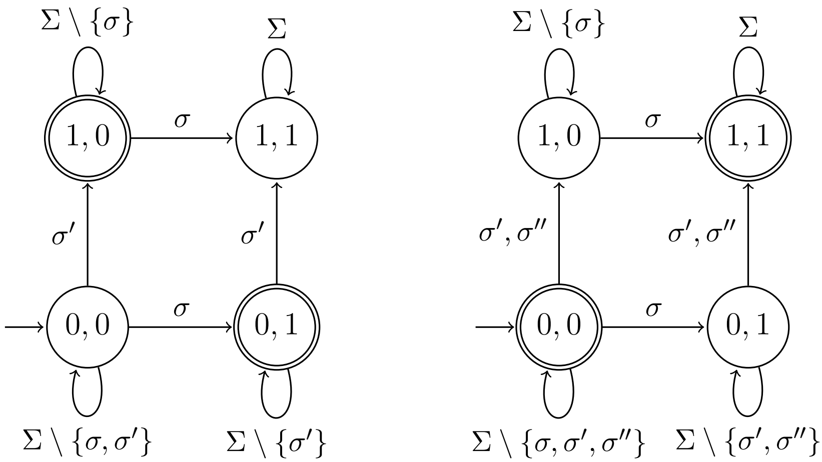
$|w|_\sigma \equiv 1 \pmod p$ where σ is the output gate. It remains to notice that the transformation monoid of each automaton is a cyclic group $\mathbb{Z}_p$ and is therefore in X.

Contains an aperiodic monoid: Assume X contains an aperiodic monoid. Then X 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

Figure 3.1: Automata for $\neg$ and $\vee$ gates when $U_1 \in X$.



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 3.1.

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 X is closed under finite direct products, this completes the proof. $\square$

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

We now consider the problem $\mathsf{AutoInt}_1(\bigcup^2 X)$. The complexity of this problem is left open in a preliminary version of the present work (Blondin & McKenzie 2012). From Proposition 2.4, $\mathsf{AutoInt}_1(\bigcup^2 X)$ generalizes $\mathsf{AutoInt}_2(X)$, however the NP-hardness from Theorem 3.15 does not apply to $\mathsf{AutoInt}_1(\bigcup^2$ Elementary abelian 2-groups). Therefore the complexity remains open in this particular case. We show the problem to be NP-hard from this more general theorem:

THEOREM 3.17. *Let X be a non trivial monoid pseudovariety, then* $\mathsf{AutoInt}_1(\bigcup^2 X)$ *is hard for* NP *under* $\leq^m_{\mathrm{AC}^0}$ *reducibility.*

PROOF. We proceed as in Theorem 3.15 by reducing from CIRCUIT–SAT. We assume that the given circuit only has negation gates at its first layer under the inputs and $\wedge, \vee$ everywhere else (this is without loss of generality, exploiting double rail logic as in the proof that the monotone circuit value problem is P-complete (Goldschlager 1977)).

Recall that if X is a non trivial pseudovariety, then either X contains an aperiodic monoid, or it contains a cyclic group $\mathbb{Z}_p$ for $p \geq 2$. We only consider the case where X contains $\mathbb{Z}_2$ since all the other cases are a direct consequence of the Theorem 3.15.

Given a circuit C, we let Σ be the set of gates of C. Each letter σ in a word accepted by all constructed automata will occur 0 or 1 times modulo 2, where 0 corresponds to false and 1 to true. For each negation gate σ with input gate σ', we build an automaton accepting words w such that $|w|_\sigma + |w|_{\sigma'} \equiv 1 \pmod 2$. For each $\vee$ gate σ with input gates σ' and σ'', we build automata accepting words w such that the following "$\vee$ formula" holds:

$$(|w|_\sigma + |w|_{\sigma'} \equiv 0 \pmod 2) \vee (|w|_\sigma + |w|_{\sigma''} \equiv 0 \pmod 2)\,.$$

For each $\wedge$ gate σ with input gates σ' and σ'', we build automata accepting words w such that the following "$\wedge$ formula" holds:

$$((|w|_\sigma + |w|_{\sigma'} \equiv 0 \pmod 2) \wedge (|w|_\sigma + |w|_{\sigma''} \equiv 0 \pmod 2)) \quad \vee \\ (|w|_\sigma \equiv 0 \pmod 2)\,.$$

We build one last automaton verifying that the output gate takes the value 1.

As shown in Table 3.2, these formulas are satisfied by an assignment iff the assignment is consistent with the semantics of the $\wedge, \vee$ gates, except in three specific cases. Therefore, our automata implement the correct semantics across circuit gates in general but mistakenly tolerate $0 \vee 1 = 0$, $1 \vee 0 = 0$ and $1 \wedge 1 = 0$. We will show that this doesn't matter since the circuit is monotone under the first layer of negation gates.

We show that C is satisfiable iff there exists a word accepted by every automaton.

Table 3.2: Validity of the formulas for $\vee$ and $\wedge$ gates. The three errors appear in bold.

$\sigma'\sigma''\sigma$	validity of	
	$\sigma' \vee \sigma'' = \sigma$	$\vee$ formula
000	✓	✓
001	✗	✗
010	**✗**	**✓**
011	✓	✓
100	**✗**	**✓**
101	✓	✓
110	✗	✗
111	✓	✓

$\sigma'\sigma''\sigma$	validity of	
	$\sigma' \wedge \sigma'' = \sigma$	$\wedge$ formula
000	✓	✓
001	✗	✗
010	✓	✓
011	✗	✗
100	✓	✓
101	✗	✗
110	**✗**	**✓**
111	✓	✓

$\Rightarrow$) Let $x_1, x_2, \ldots, x_m$ be a satisfying assignment to the gates $\sigma_1, \sigma_2, \ldots, \sigma_m$ of C, and let $w = \sigma_1^{x_1}\sigma_2^{x_2}\cdots\sigma_m^{x_m}$. As shown in Table 3.2, every valid computation in the circuit is accepted by the automata, thus w must be accepted.

$\Leftarrow$) Let w be accepted by all the automata constructed from C, leaving out the last automaton (constraining the output gate). Consider evaluating C when its input gates $\sigma_1, \sigma_2, \ldots, \sigma_i$ are assigned $|w|_{\sigma_1} \bmod 2, |w|_{\sigma_2} \bmod 2, \ldots,$ $|w|_{\sigma_i} \bmod 2$. We prove the following by induction on d:

Claim: If $|w|_\sigma$ is odd for a gate σ at depth $\leq d$, then σ evaluates to *true*.

Proof of claim: If $|w|_\sigma$ is odd for σ at depth 0 then σ was assigned *true*. So let $|w|_\sigma$ be odd for a gate σ at depth $d > 0$. If σ is a $\neg$, then the gate σ' input to σ was a circuit input gate and $|w|_{\sigma'} + |w|_\sigma$ is odd by construction, hence $|w|_{\sigma'}$ is even, so σ' was assigned *false* in C and σ evaluates to *true* as required. Otherwise, let the inputs to σ be σ' and σ''. If σ is an $\wedge$, then consider the unique row fulfilling the "$\wedge$ formula" when σ is 1 in Table 3.2. This row is the 111 row. Hence the "$\wedge$ formula" forces $|w|_{\sigma'}$ and $|w|_{\sigma''}$ to be odd. By induction, σ' and σ'' therefore evaluate to *true* in C, so that indeed σ evaluates to *true* as well. If σ is an $\vee$, then only the rows 011, 101 and 111 in Table 3.2 fulfill the "$\vee$ formula" when σ is 1. Hence either $|w|_{\sigma'}$ is odd or $|w|_{\sigma''}$ is odd. By induction, either

σ' or σ'' therefore evaluates to *true* in C, so that σ evaluates to *true*. This proves the claim.

When w is accepted by all the automata, including the last automaton forcing $|w|_{\text{output gate}}$ to be odd, the claim ensures that the output gate evaluates to *true* on the boolean assignment to the input gates induced by w.

It remains to note that the conjunction of the formulas may be expressed as the intersection of unions of two automata each with one final state. This is straightforward for the negation, output and $\vee$ gates. For the $\wedge$ formula, we use distributivity to obtain the equivalent formula

$$((|w|_\sigma + |w|_{\sigma'} \equiv 0 \pmod 2) \vee (|w|_\sigma \equiv 0 \pmod 2)) \wedge$$
$$((|w|_\sigma + |w|_{\sigma''} \equiv 0 \pmod 2) \vee (|w|_\sigma \equiv 0 \pmod 2))\,.$$

$\square$

Corollary 3.18. $\mathsf{AutoInt}_1(\bigcup^2$*Elementary abelian p-groups*) *for every* $p \geq 2$, $\mathsf{AutoInt}_1(\bigcup^2$ *Abelian groups*), $\mathsf{AutoInt}_1(\bigcup^2$*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 examine the case of $\mathsf{AutoInt}_1(\bigcup^2)$ 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 3.19. *(Knuth 1981, see p. 277 ex. 3) Let* $a_1, a_2, \ldots, a_k \in \mathbb{N}$ *and* $q_1, q_2, \ldots, q_k \in \mathbb{N}$. *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 3.20. $\mathsf{AutoInt}_1(\bigcup^2$ *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 $V[i,x] = \{v \in \mathbb{Z}_{\mathrm{ord}_{i,x}(\sigma_1)} \times \mathbb{Z}_{\mathrm{ord}_{i,x}(\sigma_2)} \times \cdots \times \mathbb{Z}_{\mathrm{ord}_{i,x}(\sigma_s)} : \sigma_1^{v_1}\sigma_2^{v_2}\cdots\sigma_s^{v_s} \in \mathrm{L}(A[i,x])\}$ 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 $\sigma_1, \sigma_2, \ldots, \sigma_s$. Thus, we have the following equivalences:

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

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 3.19. 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 3.21. $\mathsf{2\text{–}SAT} \leq^m_{\mathrm{NC}^1} \mathsf{AutoInt}_1(\bigcup^2$ *Abelian groups with* $|\Sigma| = 1)$.

Proof. Let $C(x)$ be the Boolean expression $\bigwedge_{i=1}^k C_i(x)$ over $x_1, x_2, \ldots, x_m$ 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 $p_1, p_2, \ldots, p_m$. The remainder of such an integer mod p_i represents the value of the i^{th} variable. Let

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

The language $E_1 \cap E_2 \cap \cdots \cap E_m$ 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 $(E_1 \cap E_2 \cap \cdots \cap E_m) \cap (X_1 \cap X_2 \cap \cdots \cap X_k) \neq \emptyset$ iff C is satisfiable. $\square$

Corollary 3.22. *$\mathsf{AutoInt}_1(\bigcup^2$ 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 $\mathsf{2\text{–}\oplus SAT}$ is defined similarly to $\mathsf{2\text{–}SAT}$ but with $\oplus$ operators instead of $\vee$. It is L-complete under NC^1 reducibility by (Cook & McKenzie 1987; Jones *et al.* 1976; Reingold 2005).

Theorem 3.23. $\mathsf{AutoInt}_2$(*Tight abelian group automata*) $\leq^m_{\log}$ $\mathsf{2\text{–}\oplus SAT}$.

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

Table 3.3: 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

Theorem 3.20. 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 3.20 notation). We denote this common order by $\text{ord}_i(\sigma_j)$. 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

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

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 3.3.

□

Note that we may modify the proof of Theorem 3.21 to obtain a reduction from 2–$\oplus$SAT to $\mathsf{AutoInt}_2$(Abelian groups with $|\Sigma| = 1$), therefore we obtain the following corollary.

Corollary 3.24. $\mathsf{AutoInt}_2$(*Tight abelian group automata*) *and* $\mathsf{AutoInt}_2$(*Abelian groups with* $|\Sigma| = 1$) *are* L-complete *under* $\leq^m_{\text{NC}^1}$ *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 (Garey & Johnson 1979) may be obtained in a similar fashion to Theorem 3.21.

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 3.25. $\mathsf{AutoInt}_3$(*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 Proposition 2.2 and Proposition 2.3, since their Memb counterparts are NP-complete (Beaudry 1988a,b; Beaudry *et al.* 1992).

PROPOSITION 4.1 (Beaudry 1988a,b; Beaudry *et al.* 1992). 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 (Beaudry *et al.* 1992), for solving $\mathsf{Memb}(\mathbf{J_1})$, 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 $G = \{g_1, g_2, \ldots, g_k\}$ 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 $f = h_1h_2 \cdots h_l$, since $b_i^{h_j} = i^{fh_j} = i^f = b_i$ for every $i \in [r]$. Thus, $i^{af} = i^{a(h_1h_2\cdots h_l)} = i^a$ 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. $\square$

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 (Beaudry 1988b).

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-states 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 (Beaudry *et al.* 1992), 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 (Beaudry *et al.* 1992, Fig. 1).

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

Acknowledgements

We thank the referees for helpful comments and for pointing out the need to relax the labelling of the elements transported in the definition of the $\mathsf{PT}(X)$ problem.

This work is supported by the Natural Sciences and Engineering Research Council of Canada, by the Fonds québécois de la recherche sur la nature et les technologies, by the (French) Centre national de la recherche scientifique, and by the Chaire DIGITEO ENS Cachan-École Polytechnique held by Pierre McKenzie.

References

E. Allender & M. Ogihara (1996). Relationships Among PL, #L, and the Determinant. In *RAIRO - Theoretical Information and Application*, 267–278.

V. Arvind & T. C. Vijayaraghavan (2010). Classifying Problems on Linear Congruences and Abelian Permutation Groups Using Logspace Counting Classes. *Computational Complexlity* **19**, 57–98. ISSN 1016-3328.

L. Babai, E. M. Luks & A. Seress (1987). Permutation groups in NC. In *Proc. 19th annual ACM symposium on Theory of computing*, 409–420. ISBN 0-89791-221-7.

S. Bala (2002). Intersection of Regular Languages and Star Hierarchy.

In *Proc. 29th International Colloquium on Automata, Languages and Programming*, 159–169. ISBN 3-540-43864-5.

David A. Mix Barrington (1989). Bounded-Width Polynomial-Size Branching Programs Recognize Exactly Those Languages in NC^1. *J. Comput. Syst. Sci.* **38**(1), 150–164.

David A. Mix Barrington, Neil Immerman & Howard Straubing (1990). On Uniformity within NC^1. *J. Comput. Syst. Sci.* **41**(3), 274–306.

David A. Mix Barrington & Denis Thérien (1988). Finite monoids and the fine structure of NC^1. *J. ACM* **35**(4), 941–952.

M. Beaudry (1988a). Membership testing in commutative transformation semigroups. *Information and Computation* **79**(1), 84–93. ISSN 0890-5401.

M. Beaudry (1988b). *Membership testing in transformation monoids.* Ph.D. thesis, McGill University.

M. Beaudry, P. McKenzie & D. Thérien (1992). The Membership Problem in Aperiodic Transformation Monoids. *J. ACM* **39**(3), 599–616.

Christoph Behle & Klaus-Jörn Lange (2006). FO[<]-Uniformity. In *IEEE Conference on Computational Complexity*, 183–189.

M. Blondin & P. McKenzie (2012). The complexity of intersecting finite automata having few final states. In *Proc. 7th International Computer Science Symposium in Russia*, 31–42. Springer Berlin Heidelberg.

Allan Borodin (1977). On Relating Time and Space to Size and Depth. *SIAM J. Comput.* **6**(4), 733–744.

G. Buntrock, C. Damm, U. Hertrampf & C. Meinel (1992). Structure and importance of logspace-MOD class. *Theory of Computing Systems* **25**, 223–237. ISSN 1432-4350.

K. Conrad (2013). *Characters of Finite Abelian Groups.* Lecture Notes. Available at `http://www.math.uconn.edu/~kconrad/blurbs/grouptheory/charthy.pdf`.

S. A. Cook & P. McKenzie (1987). Problems Complete for Deterministic Logarithmic Space. *J. Algorithms* **8**(3), 385–394.

M. L. Furst, J. E. Hopcroft & E. M. Luks (1980). Polynomial-Time Algorithms for Permutation Groups. In *Proc. 21st Annual Symposium on Foundations of Computer Science*, 36–41.

Z. Galil (1976). Hierarchies of complete problems. *Acta Informatica* **6**, 77–88. ISSN 0001-5903.

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

L. M. Goldschlager (1977). The Monotone and Planar Circuit Value Problems Are Log Space Complete for P. *SIGACT News* **9**(2), 25–29.

U. Hertrampf, S. Reith & H. Vollmer (2000). A note on closure properties of logspace MOD classes. *Information Processing Letters* **75**, 91–93. ISSN 0020-0190.

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

N. D. Jones, Y. E. Lien & W. T. Laaser (1976). New problems complete for nondeterministic log space. *Theory of Computing Systems* **10**, 1–17. ISSN 1432-4350.

G. Karakostas, R. J. Lipton & A. Viglas (2003). On the complexity of intersecting finite state automata and NL versus NP. *Theoretical Computer Science* **302**(1-3), 257–274. ISSN 0304-3975.

D.E. Knuth (1981). *The art of computer programming: seminumerical algorithms*, volume 2. Addidon-Wesley, 2nd edition. ISBN 9780201038224.

Michal Koucký, Pavel Pudlák & Denis Thérien (2005). Bounded-depth circuits: separating wires from gates. In *STOC*, 257–265.

D. Kozen (1977). Lower bounds for natural proof systems. In *Proc. 18th Annual Symposium on Foundations of Computer Science*, 254–266. ISSN 0272-5428.

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

E. M. Luks (1986). Parallel Algorithms for Permutation Groups and Graph Isomorphism. In *Proc. 27th Annual Symposium on Foundations of Computer Science*, 292–302.

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

E. M. Luks & P. McKenzie (1988). Parallel Algorithms for Solvable Permutation Groups. *Journal of Computer and System Sciences* **37**(1), 39–62.

B. Luong (2009). *Fourier Analysis on Finite Abelian Groups*. Birkhäuser. ISBN 9780817649159.

P. McKenzie & S. A. Cook (1987). The parallel complexity of Abelian permutation group problems. *SIAM Journal on Computing* **16**, 880–909. ISSN 0097-5397.

K. Mulmuley (1987). A fast parallel algorithm to compute the rank of a matrix over an arbitrary field. *Combinatorica* **7**, 101–104. ISSN 0209-9683.

Jean-Eric Pin (1986). *Varieties of Formal Languages*. Plenum Press.

O. Reingold (2005). Undirected ST-connectivity in log-space. In *Proc. 37th annual ACM symposium on Theory of computing*, 376–385. ISBN 1-58113-960-8.

W. J. Savitch (1970). Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences* **4**(2), 177–192. ISSN 0022-0000.

Marcel Paul Schützenberger (1965). On Finite Monoids Having Only Trivial Subgroups. *Information and Control* **8**(2), 190–194.

Howard Straubing (1994). *Finite Automata, Formal Logic and Circuit Complexity*. Birkhauser.

Pascal Tesson & Denis Thérien (2005). Complete Classifications for the Communication Complexity of Regular Languages. *Theory Comput. Syst.* **38**(2), 135–159.

Pascal Tesson & Denis Thérien (2007). Logic Meets Algebra: the Case of Regular Languages. *Logical Methods in Computer Science* **3**(1).

Denis Thérien & Thomas Wilke (1998). Over Words, Two Variables Are as Powerful as One Quantifier Alternation. In *Symposium on Theory of Computing*, 234–240.

H. Vollmer (1999). *Introduction to Circuit Complexity – A Uniform Approach*. Texts in Theoretical Computer Science. Springer Verlag.

H. T. Wareham (2001). The Parameterized Complexity of Intersection and Composition Operations on Sets of Finite-State Automata. In *Implementation and Application of Automata*, volume 2088, 302–310.

H.J. Zassenhaus (1999). *The Theory of Groups*. Dover Books on Mathematics. Dover Publications. ISBN 9780486409221.

Michael Blondin
Département d'informatique et de recherche opérationnelle
Université de Montréal
Montréal, Québec, Canada
blondimi@iro.umontreal.ca
http://www-etud.iro.umontreal.ca/~blondimi

Laboratoire Spécification et Vérification
ENS Cachan
Cachan, France
mblondin@lsv.ens-cachan.fr

Andreas Krebs
Wilhelm-Schickard-Institut für Informatik
Universität Tübingen
Tübingen, Germany
mail@krebs-net.de
http://fuseki.informatik.uni-tuebingen.de/de/mitarbeiter/krebs

Pierre McKenzie
Département d'informatique et de recherche opérationnelle
Université de Montréal
Montréal, Québec, Canada
mckenzie@iro.umontreal.ca
http://www.iro.umontreal.ca/~mckenzie

Laboratoire Spécification et Vérification
ENS Cachan
Cachan, France