The Complexity of Intersecting Finite Automata Having Few Final States

Michael Blondin and Pierre McKenzie

Département d'informatique et de recherche opérationnelle,
Université de Montréal, Montréal, QC, Canada H3T 1J4
{blondimi,mckenzie}@iro.umontreal.ca

Abstract. The problem of determining whether several finite automata accept a common word is closely related to the well-studied membership problem in transformation monoids. We review the complexity of the intersection problem and raise the issue of limiting the number of final states in the automata involved. In particular, we consider commutative automata with at most two final states and we partially elucidate the complexity of their intersection nonemptiness and related problems.

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 $i \in [k]$?

As we note in Prop. 2.1, PS and $\mathsf{AutoInt}$ are indeed equivalent in terms of their complexity, even 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. 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, AutoInt has received less attention. This is (or might be) due to the fact that it is equivalent to the membership problem when both are intractable, but appears harder to solve than the membership problem when the latter is efficiently solvable. For example, Beaudry [Bea88b] shows AutoInt for abelian groups and AutoInt for idempotent commutative monoids to be NP-complete. Beaudry points out that those two cases are examples where the automata intersection problem seems strictly harder than the transformation monoid membership problem (whose complexity is NC^3 for abelian groups and AC^0 for idempotent commutative monoids). Moreover, early results from [Gal76] show that the problem is NP-complete even when Σ is a singleton.

Nonetheless, interesting results arise from the study of AutoInt. 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 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 AutoInt are surveyed in [HK11].

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 transition monoids of the automata, or the number of generators (alphabet size), or by bounding 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 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. We obtain the partial classification of PS for abelian groups given by Table 1 below.

Table 1. Complexity of the point-spread problem for abelian groups

	Maximum size of S_i for every $i \in [m]$		
	1	2	3+
Single generator	L-complete	L-complete	NP-complete
Elementary 2-groups	$\oplus$L-complete	$\oplus$L-complete	NP-complete [Bea88b]
Elementary p-groups	Mod$_p$L-complete	? ($\in$ NP)	NP-complete [Bea88b]
General case	NC^3, FL$^{\text{ModL}}$/poly	? ($\in$ NP)	NP-complete [Bea88b]

The first line of Table 1 gives the complexity of AutoInt for abelian groups with $|\Sigma| = 1$, but it applies also to all 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 the automata intersection problem. Moreover, it provides characterizations of Mod$_p$L and thus allows the study of (some) log-space counting classes in terms of automata. The case of two final states, although incomplete, is of special interest since it appears to be efficiently solvable and generalizes group theory and linear algebra problems.

We also introduce a generalization of 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.

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

2 Preliminaries

2.1 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 Language(A). The number of occurrences of σ in a word w is denoted by $|w|_\sigma$. Throughout the paper, the automata always share the same alphabet 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 $w = w_1 \cdots w_\ell$, $T_w = T_{w_\ell} \circ \cdots \circ T_{w_1}$. 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 $\Sigma = \{a\}$, the 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. We use the notation $\leq^m_{\log}$ for log-space many-one reductions and $\leq^T_{\mathrm{NC}^1}$ for NC^1 Turing reductions. Equivalences are defined analogously and denoted by $\equiv$. See [MC87] for more details.

A function f is in GapL iff f is logspace many-one reducible to computing the integer determinant of a matrix [ABO99]. A language S is in $\mathrm{Mod}_k\mathrm{L}$ [BDHM92] iff there exists $f \in \#\mathrm{L}$ such that $x \in S \Leftrightarrow f(x) \not\equiv 0 \pmod{k}$. A language S is in ModL [AV10] iff there exists $f \in \mathrm{GapL}, g \in \mathrm{FL}$ 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 $\mathrm{FL}^{\mathrm{ModL}} = \mathrm{FL}^{\mathrm{GapL}}$ [AV10].

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.

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. $|q| \leq n$).

2.2 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, 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.

$\mathsf{PS}_b(X)$ (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. $\mathsf{AutoInt}_b(X) \equiv^m_{\mathrm{NC}^1} \mathsf{PS}_b(X)$ *for any finite monoid variety* X.

Proof. $\mathsf{AutoInt}_b(X) \leq^m_{\mathrm{NC}^1} \mathsf{PS}_b(X)$: 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 $\mathcal{M}(A_1) \times \cdots \times \mathcal{M}(A_k)$.

$\mathsf{PS}_b(X) \leq^m_{\mathrm{NC}^1} \mathsf{AutoInt}_b(X)$: 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. $\mathsf{Memb}(X) \leq^m_{\mathrm{NC}^1} \mathsf{PT}(X) \equiv^m_{\mathrm{NC}^1} \mathsf{PS}_1(X)$ *and* $\mathsf{ST}(X) \leq^m_{\mathrm{NC}^1} \mathsf{PS}(X)$.

Proof. We use the same generators for every reduction. For $\mathsf{Memb}(X) \leq^m_{\mathrm{NC}^1} \mathsf{PT}(X)$, we let $b_i = i^g$ for every $i \in [m]$ where g is the given test transformation. For $\mathsf{PT}(X) \leq^m_{\mathrm{NC}^1} \mathsf{PS}_1(X)$, we let $S_i = \{b_i\}$ for every $i \in [r]$ and $S_i = [m]$ otherwise. For $\mathsf{PS}_1(X) \leq^m_{\mathrm{NC}^1} \mathsf{PT}(X)$, 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 $\mathsf{ST}(X) \leq^m_{\mathrm{NC}^1} \mathsf{PS}(X)$, 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 execute 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

In this section we consider groups. We first record that PS_1(Groups) $\in$ NC, owing to a slick parallel reduction [Luk90, p. 27] from $\mathsf{PT}(X)$ to the problem of computing pointwise stabilizers, also known [BLS87] to be in NC. It follows that PS(Groups) is in NP by Propositions 2.2 and 2.3, and complete for NP by the forthcoming Theorem 3.12.

Proposition 3.1. PS_1(*Groups*) $\in$ NC *and* PS(*Groups*) *is* NP-complete.

We have been unable so far to solve PS_2(Groups). 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 show that $\mathsf{AutoInt}_1$(Abelian groups) $\leq$ $\mathsf{LCONNULL}$. Such a reduction can be extracted from [MC87]. However, we sketch a direct reduction, not considered explicitly in [MC87], here:

Proposition 3.2 ([MC87]). $\mathsf{AutoInt}_1$(*Abelian groups*) $\leq^T_{\mathrm{NC}^1}$ $\mathsf{LCONNULL}$.

Proof. Let $\sigma \in \Sigma$ and $\mathrm{ord}_i(\sigma)$ be the order of σ in A_i. Let $\Phi_i = \{v \in \mathbb{Z}^s_{q_i} : T_{\sigma_1^{v_1} \cdots \sigma_s^{v_s}}(\alpha_i) = \alpha_i\}$ where $q_i = \mathrm{lcm}(\mathrm{ord}_i(\sigma_1), \ldots, \mathrm{ord}_i(\sigma_s))$ and α_i is the initial state of A_i. Let B_i be the matrix such that each line is a vector from a generating set for $\Phi_i^\perp = \{v \in \mathbb{Z}^s_{q_i} : \forall u \in \Phi_i,\ u{\cdot}v = 0\}$, which may be obtained by computing a generating set for Φ_i and then calling an oracle for $\mathsf{LCONNULL}$. A generating set for Φ_i may be computed in logarithmic space because checking the existence of a word $w \in \Sigma^*$ accepted by an abelian group automaton such that $|w|_{\sigma_i} = c_i$ for every $1 \leq i \leq k$ can be done by testing accessiblity in an undirected graph. Let β_i be the final state of A_i, and $b_i = B_i x_i$ where $x_i = (|w_i|_{\sigma_1} \bmod q_i, \ldots, |w_i|_{\sigma_s} \bmod q_i)$ for any word w_i such that $T_{w_i}(\alpha_i) = \beta_i$. Then, there is a word accepted by every automaton iff this instance of LCON is feasible:

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

□

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

By Proposition 2.2, Memb(Abelian groups) $\leq^m_{\mathrm{NC}^1}$ $\mathsf{AutoInt}_1$(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 3.4. $\mathsf{AutoInt}_1$(*Elementary abelian p-groups*) *is* $\mathrm{Mod}_p\mathrm{L}$-complete.

Proof (sketch). In an elementary abelian p-group, every (nontrivial) element has order p. Therefore $\mathrm{lcm}(\mathrm{ord}_i(\sigma_1), \ldots, \mathrm{ord}_i(\sigma_s)) = p$ (or 1). Thus, the reduction from Proposition 3.2 may be converted to a log-space computable reduction to $\mathsf{LCONNULL}_p$ without any significant modification. A log-space reduction from LCON_p is also easily obtained by mapping each equation to an automaton. Since LCON_p and $\mathsf{LCONNULL}_p$ are both $\mathrm{Mod}_p\mathrm{L}$-complete [BDHM92], and $\mathrm{Mod}_p\mathrm{L} = \mathrm{Mod}_p\mathrm{L}^{\mathrm{Mod}_p\mathrm{L}}$ ($\mathrm{FMod}_p\mathrm{L} = \mathrm{FL}^{\mathrm{Mod}_p\mathrm{L}}$) [HRV00], we obtain the desired result. □

We now give the first result of this paper concerning the intersection problem with each automaton having 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 3.5. $\mathsf{AutoInt}_2$(*Elementary abelian* 2*-groups*) *is* $\oplus$L-complete.

Proof (sketch). We give a reduction to $\mathsf{LCONNULL}_2$. By following the proof of Proposition 3.2, we can derive a system of linear congruences $(B_i x \equiv b_i \pmod 2) \vee (B_i x \equiv b'_i \pmod 2)$ for every $i \in [k]$, feasible if and only if there is a word accepted by every automaton. The $\vee$-clauses are removed by introducing additional variables. More formally, we build $B_i x \equiv z_i b_i + z'_i b'_i \pmod 2$ for every $i \in [k]$ with constraints $z_i + z'_i \equiv 1 \pmod 2$ forcing the selection of either b_i or b'_i. □

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. 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 $\cap_{i=1}^{k} \cup_{j=1}^{k'} \text{Language}(A_{i,j}) \neq \emptyset$. 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 3.6. *[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 3.7. $\mathsf{AutoInt}_1(\bigcup^2$ *Tight abelian group automata*$) \leq^m_{\log}$ 2–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 $V[i,x] = \{v \in \mathbb{Z}_{\text{ord}_{i,x}(\sigma_1)} \times \cdots \times \mathbb{Z}_{\text{ord}_{i,x}(\sigma_s)} : \sigma_1^{v_1} \cdots \sigma_s^{v_s} \in \text{Language}(A[i,x])\}$ which is computable in log-space. We first note that $A[i,x]$ accepts exactly words $w \in \Sigma^*$ such that $|w|_{\sigma_j} \equiv v[i,x]_j \pmod{\text{ord}_{i,x}(\sigma_j)}$ 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:

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

$$\Leftrightarrow \exists w\, \exists x \in \{0,1\}^k \text{ such that } \bigwedge_{j=1}^{s} \left(\bigwedge_{i=1}^{k} |w|_{\sigma_j} \equiv v[i,x_i]_j \pmod{\mathrm{ord}_{i,x_i}(\sigma_j)} \right)$$

$$\Leftrightarrow \exists x \in \{0,1\}^k \text{ such that } \bigwedge_{j=1}^{s} \left(\bigwedge_{i=1}^{k} \bigwedge_{i'=1}^{k} C_{i,i',j}(x) \right),$$

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

The last equivalence is a consequence of Lemma 3.6. 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 expression. Moreover, the congruences are computable in logarithmic space since the numbers implied are tiny. □

Theorem 3.8. 2–SAT $\leq_{\log}^{m}$ $\mathsf{AutoInt}_1(\bigcup^2$ *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 3.9. $\mathsf{AutoInt}_1(\bigcup^2$ *Tight abelian group automata) and* $\mathsf{AutoInt}_1(\bigcup^2$ *Abelian groups with* $|\Sigma| = 1)$ *are* NL-complete.

Recall, that 2–⊕SAT is defined similarly to 2–SAT but with ⊕ operators instead of ∨. It is SL-complete [JLL76] and thus L-complete by SL = L [Rei05].

Theorem 3.10. $\mathsf{AutoInt}_2($*Tight abelian group automata*$)$ $\leq_{\log}^{m}$ 2–⊕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 3.7. However, it remains to show that it is possible to build an expression in 2-⊕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 3.7 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 $\wedge$. □

Corollary 3.11. $\mathsf{AutoInt}_2$(*Tight abelian group automata*) *and* $\mathsf{AutoInt}_2$(*Abelian groups with* $|\Sigma| = 1$) *are* L*-complete.*

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 3.8. 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 3.12. $\mathsf{AutoInt}_3$(*Tight abelian group automata*) *and* $\mathsf{AutoInt}_3$(*Abelian groups with* $|\Sigma| = 1$) *are* NP*-complete.*

4 Some Observations on Commutative 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, 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. □

We note that the complexity of PS($\mathbf{J_1}$) rises at least to L for two final states.

Proposition 4.3. $\mathsf{PS}_2(\mathbf{J_1})$ *is hard for* L.

Proof. We give a reduction from 2–$\oplus$SAT. For a clause $(x_1 \oplus x_2)$, we build an automaton with four states (and two final states) accepting the language $(\Sigma \setminus \{x_1, x_2\})^*(x_1x_1^* + x_2x_2^*)(\Sigma \setminus \{x_1, x_2\})^*$. For a clause $(x_1 \oplus x_2 \oplus 1)$, we build the automaton accepting the complement of the previous language. □

Unfortunately, we were not able to show $\mathsf{PS}_2(\mathbf{J_1}) \in$ L, even though it appears to be a reasonable claim. The previous hardness proof yields very specific automata, more exactly they have four states and at most one transition between any two distinct states. Only with such an outrageous restriction are we currently able to show a log-space upper bound.

For the sake of completeness, we note that the problem is NP-complete for three final states, as proved in [Bea88b].

Proposition 4.4 ([Bea88b]). $\mathsf{PS}_3(\mathbf{J_1})$ *is* NP-complete.

5 Conclusion and Further Work

The question marks in Table 1 and the lack of upper bound in Proposition 4.3 indicate that further work is needed to properly locate the complexity of the point-spread problem for two final states. Judging from our results for unary languages and elementary abelian 2-groups, we might suspect PS_2 and $\mathsf{AutoInt}_2$ to remain efficiently solvable for abelian permutation groups and for idempotent commutative monoids. However, our current proof for elementary abelian 2-groups cannot be extended. We suspect that PS_2(Elementary abelian p-groups) is harder for $p > 2$ than for $p = 2$.

It would be interesting to answer such questions since PS_2(Abelian groups) is equivalent to testing feasibility of linear congruences of the type $(B_1x \equiv b_1 \pmod{q_1} \vee B_1x \equiv b_1' \pmod{q_1}) \wedge \cdots \wedge (B_kx \equiv b_k \pmod{q_k} \vee B_kx \equiv b_k' \pmod{q_k})$. The variant of $\mathsf{AutoInt}_1$, with two automata per $\cup$-clause, yields similar linear congruences, but with B_i and q_i differing in each $\vee$-clause. Therefore, exploring $\mathsf{AutoInt}_2$ and $\mathsf{AutoInt}_2(\cup^2)$ appears to be a fruitful line of research to obtain interesting variants of PS efficiently solvable.

Finally, PS_1(Solvable groups) $\in$ NC using [Luk90, p. 27] and [BLS87]. But what about PS_2(Solvable groups) and PS_2(Groups)? An NC toolkit is available, but are the tools sufficient?

References

[ABO99] Allender, E., Beals, R., Ogihara, M.: The complexity of matrix rank and feasible systems of linear equations. Comput. Complex. 8(2), 99–126 (1999)

[AV10] Arvind, V., Vijayaraghavan, T.C.: Classifying problems on linear congruences and abelian permutation groups using logspace counting classes. Comput. Complex. 19, 57–98 (2010)

[Bal02] Bala, S.: Intersection of Regular Languages and Star Hierarchy. In: Widmayer, P., Triguero, F., Morales, R., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 159–169. Springer, Heidelberg (2002)

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

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

[Bea88b] Beaudry, M.: Membership testing in transformation monoids. PhD thesis, McGill University (1988)

[BLS87] Babai, L., Luks, E.M., Seress, A.: Permutation groups in NC. In: Proc. 19th Annual ACM Symposium on Theory of Computing, pp. 409–420 (1987)

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

[FHL80] Furst, M.L., Hopcroft, J.E., Luks, E.M.: Polynomial-time algorithms for permutation groups. In: FOCS, pp. 36–41 (1980)

[Gal76] Galil, Z.: Hierarchies of complete problems. Acta Informatica 6, 77–88 (1976)

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

[HK11] Holzer, M., Kutrib, M.: Descriptional and computational complexity of finite automata – a survey. Inf. Comput. 209(3), 456–470 (2011)

[HRV00] Hertrampf, U., Reith, S., Vollmer, H.: A note on closure properties of logspace mod classes. Inf. Process. Lett. 75, 91–93 (2000)

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

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

[Knu81] Knuth, D.E.: The art of computer programming: seminumerical algorithms, 2nd edn., vol. 2. Addison-Wesley (1981)

[Koz77] Kozen, D.: Lower bounds for natural proof systems. In: Proc. 18th Annual Symposium on Foundations of Computer Science, pp. 254–266 (1977)

[LM88] Luks, E.M., McKenzie, P.: Parallel algorithms for solvable permutation groups. J. Comput. Syst. Sci. 37(1), 39–62 (1988)

[LR92] Lange, K.-J., Rossmanith, P.: The Emptiness Problem for Intersections of Regular Languages. In: Havel, I.M., Koubek, V. (eds.) MFCS 1992. LNCS, vol. 629, pp. 346–354. Springer, Heidelberg (1992)

[Luk86] Luks, E.M.: Parallel algorithms for permutation groups and graph isomorphism. In: FOCS, pp. 292–302. IEEE Computer Society (1986)

[Luk90] Luks, E.M.: Lectures on polynomial-time computation in groups. Technical report. College of Computer Science, Northeastern University (1990)

[MC87] McKenzie, P., Cook, S.A.: The parallel complexity of abelian permutation group problems. SIAM J. Comput. 16, 880–909 (1987)

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

[Rei05] Reingold, O.: Undirected st-connectivity in log-space. In: Proc. 37th Annual ACM Symposium on Theory of Computing, pp. 376–385 (2005)

[Sav70] Savitch, W.J.: Relationships between nondeterministic and deterministic tape complexities. J. Comput. Syst. Sci. 4(2), 177–192 (1970)

[War01] Wareham, H.T.: The Parameterized Complexity of Intersection and Composition Operations on Sets of Finite-State Automata. In: Yu, S., Păun, A. (eds.) CIAA 2000. LNCS, vol. 2088, pp. 302–310. Springer, Heidelberg (2001)