Equivalence Testing of Weighted Automata over Partially Commutative Monoids[*]

V. Arvind
Institute of Mathematical Sciences (HBNI), Chennai,India

Abhranil Chatterjee
Institute of Mathematical Sciences (HBNI), Chennai, India

Rajit Datta
Chennai Mathematical Institute, India

Partha Mukhopadhyay
Chennai Mathematical Institute, India

Abstract

Motivated by equivalence testing of k-tape automata, we study the *equivalence* testing of weighted automata in the more general setting, over partially commutative monoids (in short, pc monoids), and show efficient algorithms in some special cases, exploiting the structure of the underlying non-commutation graph of the monoid.

Specifically, if the edge clique cover number of the non-commutation graph of the pc monoid is a constant, we obtain a deterministic quasi-polynomial time algorithm for equivalence testing. As a corollary, we obtain the first deterministic quasi-polynomial time algorithms for equivalence testing of k-tape weighted automata and for equivalence testing of deterministic k-tape automata for constant k. Prior to this, the best complexity upper bound for these k-tape automata problems were randomized polynomial-time, shown by Worrell [24]. Finding a polynomial-time deterministic algorithm for equivalence testing of deterministic k-tape automata for constant k has been open for several years [13] and our results make progress.

We also consider pc monoids for which the non-commutation graphs have an edge cover consisting of at most k cliques and star graphs for any constant k. We obtain a randomized polynomial-time algorithm for equivalence testing of weighted automata over such monoids.

Our results are obtained by designing efficient zero-testing algorithms for weighted automata over such pc monoids.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory; Theory of computation

Keywords and phrases Weighted Automata, Automata Equivalence, Partially Commutative Monoid

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.10

Acknowledgements We thank the anonymous reviewers for their helpful feedback.

1 Introduction

Testing the equivalence of two multi-tape finite automata is a fundamental problem in automata theory. For a k-tape automaton, we denote the mutually disjoint alphabets for the k tapes by $\Sigma_1, \dots, \Sigma_k$. The automaton accepts a subset of the product monoid $\Sigma_1^* \times \cdots \times \Sigma_k^*$. Two multi-tape automata are *equivalent* if they accept the same subset.

[*] A preliminary version of this work was presented at HIGHLIGHTS of Logic, Games and Automata 2020.

Equivalence testing of multi-tape *non-deterministic* automata is undecidable [14]. The problem was shown to be decidable for 2-tape *deterministic* automata independently by Bird [5] and Valiant [22]. Subsequently, an exponential upper bound was shown for it [3]. Eventually, a polynomial-time algorithm was obtained by Friedman and Greibach [13] and the authors conjectured that equivalence testing of deterministic k-tape automata for any constant k is in polynomial time.

A closely related problem is testing the *multiplicity equivalence* of non-deterministic multi-tape automata. The multiplicity equivalence testing problem is to decide whether for each tuple in the product monoid $\Sigma_1^* \times \cdots \times \Sigma_k^*$, the number of accepting paths in the two input automata is the same. Since a deterministic automaton has at most one accepting path for each input word, the equivalence of deterministic k-tape automata coincides with multiplicity equivalence. More generally, equivalence testing for *weighted automata* (over the underlying field or ring of coefficients) is to decide if the coefficient of each word (i.e. the total sum of weights of each accepting path) is the same for the two given automata. For the weighted case, equivalence testing is in deterministic polynomial time for one-tape automata [19, 21]. Equivalence testing of k-tape weighted automata was shown *decidable* by Harju and Karhumäki [15] using the theory of free groups [1]. An improved complexity-theoretic upper bound remained elusive for k-tape multiplicity equivalence testing, until recently Worrell [24] obtained a *randomized* polynomial-time algorithm for testing the equivalence of k-tape weighted automata (and equivalence testing of deterministic k-tape automata) for any constant k. Worrell takes a different approach via Polynomial Identity Testing (PIT). In [24], Worrell asked if the equivalence testing problem for k-tape weighted automata can be solved in *deterministic* polynomial time, for constant k.

This Paper. Building on Worrell's results [24] and exploiting further the connections between weighted automata equivalence and polynomial identity testing, we show that equivalence testing of two k-tape weighted automata is in *deterministic* quasi-polynomial time. This immediately yields the first deterministic quasi-polynomial time algorithm for equivalence testing of deterministic k-tape automata.

Our approach solves a more general problem in the setting of *partially commutative monoids*. To motivate this, let us consider k-tape weighted automata in this setting. The *product monoid* $M = \Sigma_1^* \times \cdots \times \Sigma_k^*$ associated with k-tape automata is a *partially commutative monoid* (henceforth, *pc monoid*), in the sense that any two variables $x \in \Sigma_i, y \in \Sigma_j, i \neq j$ commute with each other [2]. Variables in the same tape alphabet Σ_i are mutually non-commuting. We associate a *non-commutation graph* G_M with M to describe the non-commutation relations: (x, y) is an edge if and only if x and y do not commute. If there is no edge (x, y) in G_M, the words xy and yx are equivalent as the variables x and y commute. The words over any pc monoid are defined with respect to the equivalence relation induced by the non-commutation graph of the pc monoid. The notion of words and their equivalence over a pc monoid is formally explained in Section 3. For the k-tape case, the non-commutation graph G_M is a union of k disjoint cliques: its vertex set is $\Sigma_1 \cup \ldots \cup \Sigma_k$ and G_M is the union of k disjoint cliques, induced by each Σ_i.

More generally, we obtain an equivalence testing algorithm for weighted automata over any pc monoid whose non-commutation graph has a constant-size edge clique cover (*not necessarily* disjoint) with a constant number of isolated vertices. Recall that the edge clique

[1] This also shows the decidability of equivalence problem for deterministic multi-tape automata.
[2] These are sometimes also called as free partially commutative monoids.

cover of a graph is a collection of subgraphs where each subgraph is a clique and each edge of the graph is contained in at least one of the subgraphs. The size of the edge clique cover is the number of cliques in it.

The isolated vertices can be thought of as a part of the edge clique cover by adding a new vertex (variable) for each isolated vertex and introducing a matching edge between them. Henceforth, we will not worry about the isolated vertices separately and consider them as part of the edge clique cover. We call such monoids as k-*clique monoids* where the edge clique cover size is bounded by k.

▶ Remark 1.1. Since two weighted automata, A and B are equivalent if and only if their difference $C = A - B$ is a weighted automaton equivalent to zero (formally explained in Section 2), we can describe the results in terms of zero-testing of a weighted automaton.[3]

In this paper, the field $\mathbb{F}$ from which the weights of automata are taken is an infinite field. For computational implementation, we assume that the field arithmetic can be performed efficiently (for example, $\mathbb{F}$ could be the field of rational numbers). Also, throughout the paper the size of an automaton refers to the number of states.

▶ **Theorem 1.2.** *Let A be an input $\mathbb{F}$-weighted automaton of size s over a pc monoid M such that its non-commutation graph G_M has an edge clique cover of size k. Then, the zero-testing of A has a deterministic $(nks)^{O(k^2 \log ns)}$ time algorithm. Here n is the size of the alphabet of M, and the edge clique cover is given as part of the input.*

It is interesting to note that the the decidability of the equivalence problem over partially commutative monoids is already studied [23]. As an immediate corollary, the above theorem yields a deterministic quasi-polynomial time algorithm for equivalence testing of k-tape weighted automata (also for equivalence testing of deterministic k-tape automata). Notice that, for the k-tape case, the edge clique cover of size k is also part of the input since for each $1 \leq i \leq k$, the i^{th} tape alphabet Σ_i is explicitly given and it induces a clique.

▶ **Corollary 1.3.** *The equivalence testing problem for k-tape weighted automata and deterministic k-tape automata can be solved in deterministic quasi-polynomial time for constant k.*

Next, we consider equivalence testing over more general pc monoids M.

Given a graph $G = (\mathrm{X}, E)$, a collection of k graphs $\{G_i = (\mathrm{X}_i, E_i)\}_{i=1}^{k}$ such that $\mathrm{X} = \cup_{i=1}^{k} \mathrm{X}_i$ and $E = \cup_{i=1}^{k} E_i$ is called a k-*covering* of G. It seems natural to investigate whether there are covers other than just edge clique cover for which one can obtain efficient equivalence test.

We say M is a k-*monoid* if its non-commutation graph G_M has a 2-covering $\{G_1, G_2\}$ such that, for some $k' \leq k$, G_1 has an edge clique cover of size at most k' and G_2 has a vertex cover of size at most $k - k'$ (hence the edges of G_2 can be covered by $k - k'$ many star graphs). We show that equivalence testing over k-monoids has a randomized polynomial-time algorithm for constant k. This result can be seen as a generalization of Worrell's result [24].

▶ **Theorem 1.4.** *Let A be an input $\mathbb{F}$-weighted automaton of size s over a k-monoid M. Then the zero-testing of A can be decided in randomized $(ns)^{O(k)}$ time. Here n is the size of the alphabet of M.*

[3] The difference C of two weighted automata A and B means the weight of each word w in C is the difference between the weights of w in A and B.

▶ Remark 1.5. What is the complexity of equivalence testing for weighted automata over an arbitrary pc monoid? The non-commutation graph G_M of any pc monoid M over the alphabet X trivially has an edge clique cover of size bounded by $\binom{|X|}{2}$. Hence, the above results would only give an exponential-time algorithm. Note that if G_M has an *induced matching* [4] of size more than k then M is not a k-monoid. Call M a *matching monoid* if G_M is a perfect matching. It follows from Lemma 3.3, shown in Section 3, that equivalence testing over arbitrary pc monoids is deterministic polynomial-time reducible to equivalence testing over matching monoids. Thus, the complexity of zero-testing of $\mathbb{F}$-weighted automata over matching monoids is essentially the most general case. We also note that Worrell has shown that the evaluation problem for multi-tape automata is #P-complete if the number of tapes is not fixed [24, Proposition 3].

Various automata-theoretic problems have been studied in the setting of pc monoids. For example, pc monoids have found applications in modeling the behavior of concurrent systems [16]. Droste and Gastin [10] have studied the relation between recognizability and rationality over pc monoids.

Proofs Overview. Our proof is inspired by Worrell's key insight [24] that the k-tape automata equivalence problem can be reduced to a suitable instance of polynomial identity testing problem over partially commuting variables. Worrell's algorithm is randomized. In contrast, since we are considering automata over general pc monoids and we aim to design efficient deterministic algorithms, we require additional ideas. First, we suitably apply a classical algebraic framework to transfer the zero-testing problem over general pc monoids to pc monoids whose non-commutation graphs are a disjoint union of cliques [6, 8]. This allows us to generalize a zero-testing criteria for weighted automata over standard noncommutative setting [11, Cor. 8.3] to the setting of *general pc monoids*. The generalization states that any nonzero weighted automata of size s over any pc monoid must have a non-zero word within the length $\mathrm{poly}(s, n)$ where n is the alphabet size. This allows us to reduce zero-testing of weighted automata to an instance of polynomial identity testing over pc monoids, where these polynomials are computable by small algebraic branching programs (ABPs) over pc monoids. Over noncommutative variables, ABPs are well-studied in arithmetic circuit complexity [17]. It turns out that we can solve the identity testing problem for ABPs over k-clique monoids in deterministic quasi-polynomial time by suitably adapting a black-box polynomial identity test for noncommutative algebraic branching programs based on a quasi-polynomial size hitting set construction [12]. Our algorithm recursively builds on this construction, ensuring that the resulting hitting set remains of quasi-polynomial size.

The proof of Theorem 1.4 is along similar lines. First, we obtain a randomized polynomial-time identity testing algorithm over pc monoids whose non-commutation graph has a k-vertex cover for constant k. This algorithm itself uses ideas from automata theory. Then a composition lemma yields an identity testing algorithm over k-monoids.

The paper is organized as follows. In Section 2, we provide the necessary background. We prove a zero testing criteria for automata over pc monoids in Section 3. Theorem 1.2 is presented in Section 4, and Theorem 1.4 in Section 5. Some proof details are in the appendix.

[4] An induced matching is a matching that includes every edge connecting any two vertices in the subset as an induced subgraph.

2 Preliminaries

We recall basic definitions and results, mainly from automata theory and arithmetic circuit complexity, and define notations used in the paper.

Notation. Let $\mathbb{F}$ be an infinite field. $\mathrm{Mat}_t(\mathbb{F})$ denotes the ring of $t \times t$ matrices over $\mathbb{F}$. For matrices A and B of sizes $m \times n$ and $p \times q$ respectively, their tensor (Kronecker) product $A \otimes B$ is defined as the block matrix $(a_{ij}B)_{1\le i\le m, 1\le j\le n}$, and the dimension of $A \otimes B$ is $pm \times qn$. Given bases $\{v_i\}_{1\le i\le \dim(V)}$ and $\{w_j\}_{1\le j\le \dim(W)}$ for the vector spaces V and W, the vector space $V \otimes W$ is the tensor product space with a basis $\{v_i \otimes w_j\}_{1\le i\le \dim(V), 1\le j\le \dim(W)}$.

For a series (resp. polynomial) S and a word (resp. monomial) w, let $[w]S$ denote the coefficient of w in the series S (resp. polynomial). In this paper, we consider weighted automata over a field $\mathbb{F}$ and alphabet (or variables) $\mathrm{X} = \{x_1, \ldots, x_n\}$.

Arithmetic Circuit Complexity. An *algebraic branching program* (ABP) is a layered directed acyclic graph with one in-degree-0 vertex called *source*, and one out-degree-0 vertex called *sink*. Its vertex set is partitioned into layers $0, 1, \ldots, d$, with directed edges only between adjacent layers (i to $i+1$). The source and the sink are in layers zero and d, respectively. Each edge is labeled by a linear form over $\mathbb{F}$ in variables $\mathrm{X} = \{x_1, \ldots, x_n\}$. The polynomial computed by the ABP is the sum over all source-to-sink directed paths of the product of linear forms that label the edges of the path. The maximum number of nodes in any layer is called the width of the algebraic branching program. The size of the branching program is taken to be the total number of nodes.

Equivalently, the computation of an algebraic branching program can be defined via the iterated matrix product $\lambda^T M_1 M_2 \cdots M_d \mu$, where λ, μ are vectors in $\mathbb{F}^w$ and each M_i is a $w \times w$ matrix whose entries are affine linear forms over X. Here w corresponds to the ABP width and $d+1$ corresponds to the number of layers in the ABP.

If X is a set of non-commuting variables then the ABP is a noncommutative algebraic branching program (e.g., see [17]).

Let $S \subset \mathbb{F}\langle \mathrm{X} \rangle$ be a subset of polynomials in the noncommutative polynomial ring $\mathbb{F}\langle \mathrm{X} \rangle$. A mapping $\boldsymbol{v} : \mathrm{X} \to \mathrm{Mat}_t(\mathbb{F})$ from variables to $t \times t$ matrices, it defines an *evaluation map* defined for any polynomial $f \in \mathbb{F}\langle \mathrm{X} \rangle$ as $\boldsymbol{v}(f) = f(v(x_1), \ldots, v(x_n))$. A collection H of such evaluation maps is a *hitting set* for S, if for every nonzero f in S, there is an evaluation $\boldsymbol{v} \in H$ such that $\boldsymbol{v}(f) \neq 0$.

Let $S_{n,d,s}$ denote the set of noncommutative polynomials in $\mathbb{F}\langle \mathrm{X} \rangle$ (where $n = |\mathrm{X}|$) that have algebraic branching programs of size s and d layers. Forbes and Shpilka [12] have given a quasi-polynomial size hitting set $H_{n,d,s}$ computable in quasi-polynomial time for $S_{n,d,s}$ that can. Moreover, the matrix tuples in $H_{n,d,s}$ are $d+1$ dimensional.

▶ **Theorem 2.1** ([12, Theorem I.8]). *For all $s, d, n \in \mathbb{N}$, if $|\mathbb{F}| \ge \mathrm{poly}(d, n, s)$ then there is a hitting set $H_{n,d,s}$ for $S_{n,d,s}$. Further $|H_{n,d,s}| \le (sdn)^{O(\log d)}$ and $H_{n,d,s}$ is computable in deterministic time $(sdn)^{O(\log d)}$.*

Automata Theory. We recall some basic algebraic automata theory from the Berstel-Reutenauer book [4].

Let $\mathbb{F}$ be a field[5] and X be an alphabet. A $\mathbb{F}$-weighted automaton[6] A over X has a finite set of states Q. There is a weight function $E : Q \times \mathrm{X} \times Q \to \mathbb{F}$ that assigns a weight to each transition. The number of states, $|Q|$ is the *size* of the automaton. A path is a sequence

[5] In general $\mathbb{F}$ can be a semiring, but for our purpose it suffices to consider fields.
[6] Sometime called nondeterministic weighted automata in the literature.

of edges: $(q_0, x_1, q_1)(q_1, x_2, q_2)\cdots(q_{t-1}, x_t, q_t)$. The weight of the path is the product of the weights of the edges. For each word $w = x_1x_2\cdots x_t \in \mathrm{X}^*$, the coefficient of w, $[w]S$ is the total contribution of all the paths between a start and accepting state for the word w, which is an element of $\mathbb{F}$. This defines a formal series $S = \sum_{w\in \mathrm{X}^*}[w]S \cdot w$ which is an element of the *formal power series ring* $\mathbb{F}\langle\langle \mathrm{X}\rangle\rangle$. We say that S is the formal series *recognized* by the (weighted) automaton A.

Multi-tape automata. Next, we briefly explain weighted multi-tape automata defined in terms of pc monoids. Let M be the pc monoid over variables $\mathrm{X} = \mathrm{X}_1 \cup \cdots \cup \mathrm{X}_k$ defined as follows: the variables in each X_i are non-commuting, but for all $i \neq j$ and any $x \in \mathrm{X}_i, y \in \mathrm{X}_j$ we have $xy = yx$. As defined already, the transition function E is a mapping $Q \times \mathrm{X} \times Q \to \mathbb{F}$. A path is a sequence of edges : $(q_0, x_1, q_1)(q_1, x_2, q_2)\cdots(q_{t-1}, x_t, q_t)$ where each $x_i \in \mathrm{X}_j$ for some j. The label of the run is $m = x_1x_2\cdots x_t$ in the pc monoid M, and $[m]A$ is the total contribution of all the runs between start and accepting states having the label equivalent to m.

An automaton is *deterministic* if the set of states can be partitioned as $Q = Q^{(1)}\sqcup\ldots\sqcup Q^{(k)}$, where states in $Q^{(i)}$ read input only from the i^{th} tape alphabet X_i, and each state has a single transition for every input variable. Thus, a deterministic automaton has at most one accepting path for each input $m \in M$.

Now we explain how equivalence testing of weighted automata is polynomial-time reducible to zero testing of weighted automata. Let A and B be $\mathbb{F}$-weighted automata over the alphabet X. The transition matrices N_A and N_B are defined as follows: $N_A[i,j] = \sum_{x\in X} E_A(q_i, x, q_j)\cdot x$. ($N_B$ is defined similarly)[7]. Let the series computed by A and B be $\lambda_A^T \cdot \sum_{i\geq 0} N_A^i \cdot \mu_A$ and $\lambda_B^T \cdot \sum_{i\geq 0} N_B^i \cdot \mu_B$, respectively. Here $\lambda_A, \mu_A, \lambda_B, \mu_B$ are column scalar vectors. Define the weighted automaton C with transition matrix N_C and the scalar vectors λ_C, μ_C as follows:

$$\lambda_C = \begin{bmatrix}\lambda_A \\ \lambda_B\end{bmatrix}, \qquad N_C = \begin{bmatrix} N_A & 0 \\ 0 & N_B\end{bmatrix}, \qquad \mu_C = \begin{bmatrix}\mu_A \\ -\mu_B\end{bmatrix}.$$

We state an easy fact also used in [24].

▶ **Fact 1.** *A and B are equivalent if and only if C is a zero automaton.*

3 A Zero Testing Criteria Over Partially Commutative Monoids

A basic result in algebraic automata theory, says that an $\mathbb{F}$-weighted automaton A of size s represents a nonzero series in $\mathbb{F}\langle\langle \mathrm{X}\rangle\rangle$ if and only if there is a word $w \in \mathrm{X}^*$ of length at most $s-1$, such that $[w]S$ is nonzero. It has a simple linear algebraic proof [11, Corollary 8.3, Page 145][8].

In this section, we prove a theorem similar in spirit over general pc monoids.

Pc monoids and partitioned pc monoids. Let X be a finite alphabet (equivalently, variable set). Formally, a pc monoid M over X is a pair $M = (\mathrm{X}^*, I)$ where $I \subseteq \mathrm{X}\times\mathrm{X}$ be such that $(x_1, x_2) \in I$ if and only if $x_1x_2 = x_2x_1$. I is reflexive and symmetric. Let $\tilde{I}$ be the *congruence* generated from I by transitive closure. The monoid elements are defined as the congruence classes $\tilde{m}$ for $m \in \mathrm{X}^*$. In other words, M is a factor monoid of X^* generated by $\tilde{I}$. The *non-commutation graph* $G_M = (\mathrm{X}, E)$ of M is a simple undirected graph such that $(x_1, x_2) \in E$ if and only if $(x_1, x_2) \notin I$.

[7] Here $q_1, q_2, \ldots$ be the enumeration of the states of A (and similarly for B).
[8] This result is generally attributed to Schützenberger.

A pc monoid M over alphabet (i.e. variable set) X is a *k-partitioned pc monoid* if its non-commutation graph G_M has a k-covering $\{G_i\}_{i=1}^k$ such that the subgraphs G_i are pairwise vertex disjoint. Given any pc monoid M with a k-covering, we can associate a k-partitioned pc monoid M' with it, such that M is isomorphic to a submonoid of M', as follows.

Suppose $G_M = (X, E)$ has a k-covering $\{G_i\}_{i=1}^k$, where $G_i = (X_i, E_i)$. Let $\hat{X} = \{x_{ti} \mid 1 \leq t \leq n, 1 \leq i \leq k\}$ be kn new variables. Here $|X| = n$ and $X = x_1, \ldots, x_n$. Let $G'_i = (X'_i, E'_i)$ be a copy of G_i obtained by replacing the vertex $x_t \in X_i$ by its i^{th} copy x_{ti}, such that (x_{ti}, x_{si}) is an edge in G'_i if and only if (x_t, x_s) is an edge in G_i. Let G' denote the disjoint union graph $G' = G'_1 \sqcup G'_2 \sqcup \cdots \sqcup G'_k$, and M' be the pc monoid whose non-commutation graph is $G' = (X', E')$. Clearly, M' is a k-partitioned pc monoid, defined by M and its given k-covering.

As an $\mathbb{F}$-algebra, we note that $\mathbb{F}\langle M'\rangle$ is isomorphic to the tensor product of the $\mathbb{F}$-algebras $\mathbb{F}\langle M'_1\rangle \otimes \cdots \otimes \mathbb{F}\langle M'_k\rangle$ where M'_i is the pc monoid defined by G'_i.

The following simple observation, which shows that the pc monoid M is isomorphic to a submonoid of M', is well-known [6, 8, 9].

▶ **Lemma 3.1.** *Let $\psi : \mathbb{F}\langle M\rangle \to \mathbb{F}\langle M'\rangle$ be the map such that $\psi(m) = m_1 \otimes m_2 \otimes \cdots \otimes m_k$ for any monomial m in M and extend by linearity. Here, for $1 \leq i \leq k$, the monomial m_i is obtained from m (by dropping the letters of m not in X_i) and replacing each occurrence $x_t \in \mathrm{X}_i$ by the variable x_{ti}, $1 \leq t \leq n$. Then, ψ is an injective homomorphism.*

▶ Remark 3.2. We include a self-contained proof in the appendix for completeness, in our notation.

By Lemma 3.1, zero testing for weighted automata over pc monoids is reducible to zero-testing of weighted automata over partitioned pc monoids in deterministic polynomial time. More formally, we show the following.

▶ **Lemma 3.3.** *Let A be the given $\mathbb{F}$-weighted automaton of size s over a pc monoid M for which the non-commutation graph G_M has k-covering $\{G_i = (\mathrm{X}_i, E_i)\}_{i=1}^k$. Then zero testing of A is reducible to the zero testing of another $\mathbb{F}$-weighted automaton B over the associated k-partitioned pc monoid M' in deterministic polynomial time. Moreover, the size of the automaton B is $O(ns^2k)$.*

Proof. The automaton B is simply obtained by applying the map ψ on the variables in M. For a variable x_t, let $J_t \subseteq \{1, 2, \ldots, k\}$ be the set of indices such that, $i \in J_t$ if and only if $x_t \in \mathrm{X}_i$. Then $\psi(x_t) = \eta_{i_1} \otimes \cdots \otimes \eta_{i_{|J_t|}}$ where $i_1 < i_2 < \cdots < i_{|J_t|}$ and for each j, $i_j \in J_t$, $\eta_{i_j} = x_{ti_j}$. Now for each $q_0, q_k \in Q$ such that $(q_0, x_t, q_k) \in E$ [9] and $wt(q_0, x_t, q_k) = \alpha \in \mathbb{F}$, we introduce new states $q_1, \ldots, q_{|J_t|-1}$ and for each $j \leq |J_t| - 1$, add the edge $e_j = (q_{j-1}, \eta_{i_j}, q_j)$ in E and $wt(e_1) = \alpha$ and for other newly added edges the weight is 1. Since the number of edges in A is $O(ns^2)$, it is easy to see the number of nodes in B is $O(ns^2k)$. The fact that A computes the zero series if and only if B computes the zero series, follows from Lemma 3.1 and in particular from the fact that ψ is injective on the set of monomials. ◀

Worrell has already proved that the zero-testing of weighted automata over partitioned monoids whose non-commutation graphs are union of disjoint cliques, can be reduced to the identity testing of noncommutative ABPs [24]. We restate a proposition from Worrell's paper in our framework.

[9] Here for simplicity of notation, we have used q_0, q_k to represent an arbitrary pair such that there is a transition between them, and q_0 is not necessarily the initial state.

▶ **Proposition 3.4** (Adaptation of Proposition 5 of [24]). *Let A be a given $\mathbb{F}$-weighted automaton of size s over a partitioned pc monoid M computing a series S. Moreover the non-commutation graph G_M is the disjoint union of k cliques. Let N be the transition matrix of A. Then S is the zero series if and only if the ABPs $\lambda^T N^\ell \mu = 0$ for each $0 \le \ell \le s-1$, where λ, μ are vectors in $\mathbb{F}^s$.*

Combining Lemma 3.3 and Proposition 3.4, we obtain the following generalization of [11, Corollary 8.3] over arbitrary pc monoids which may be of independent interest.

▶ **Theorem 3.5.** *Let A be a given $\mathbb{F}$-weighted automaton of size s over any pc monoid M representing a series S. Then S is a nonzero series if and only if there exists a word $w \in \mathrm{X}^*$ such that $[w]S$ is nonzero where the length of w is bounded by $O(n^3 s^2)$.*

Proof. Observe that the non-commutation graph G_M has a trivial edge clique cover of size $\le n^2$ where n is the size of the alphabet. Then we apply Lemma 3.3 to conclude that S is a zero series if and only if the series S' computed by the $\mathbb{F}$-weighted automaton B over the associated partitioned pc monoid (whose non-commutation graph is a disjoint union of cliques) is zero. The size s' of B is bounded by $O(n^3 s^2)$. Now we use Proposition 3.4 to see that S' is identically zero if and only if the ABPs $\lambda^T N^\ell \mu = 0$ for each $0 \le \ell \le s'-1$ are identically zero where N is the transition matrix of B. Now notice that under the image of ψ map, the length of any word can only increase. In other words, for any word $w : |\psi(w)| \ge |w|$. It follows that $(S' = \psi(S))^{\le s'-1}$ is a nonzero polynomial where S' is the part of $\psi(S)$ of degree at most $s'-1$. Since ψ is injective, it must be the case that $S^{\le s'-1}$ is also a nonzero polynomial, and the theorem follows. ◀

4 Deterministic Zero Testing of Weighted Automata Over k-Clique Monoids

In this section, we show that zero testing for weighted automata over k-clique monoids for constant k is in deterministic quasi-polynomial time. In fact, by Lemma 3.3 and Proposition 3.4, the zero testing problem reduces to the polynomial identity testing of ABPs over partitioned pc monoids whose non-commutation graph is a disjoint union of k cliques. Thus, in order to prove Theorem 1.2 it suffices to design an efficient identity testing algorithm for ABPs computing polynomials in $\mathbb{F}\langle \mathrm{X}_1\rangle \otimes \cdots \otimes \mathbb{F}\langle \mathrm{X}_k\rangle$, where k is a constant and the variable sets $\mathrm{X}_j = \{x_{ij}\}_{1\le i\le n}$ are of size n each and pairwise disjoint.

Evaluation over algebras. For a polynomial $f \in \mathbb{F}\langle \mathrm{X}_1\rangle \otimes \cdots \otimes \mathbb{F}\langle \mathrm{X}_k\rangle$ and a k-tuple of $\mathbb{F}$-algebras $\boldsymbol{A} = (A_1, \ldots, A_k)$, an *evaluation* of f in $\boldsymbol{A}$ is given by a k-tuple of maps $\boldsymbol{v} = (v_1, v_2, \ldots, v_k)$, where $v_i : \mathrm{X}_i \to A_i$. We can extend it to the map $\boldsymbol{v} : \mathbb{F}\langle \mathrm{X}_1\rangle \otimes \cdots \otimes \mathbb{F}\langle \mathrm{X}_k\rangle \to A_1 \otimes \cdots \otimes A_k$ as follows: For any monomial $m = m_1 \otimes \cdots \otimes m_k$ where $m_i \in \mathrm{X}_i^*$, let $\boldsymbol{v}(m) = v_1(m_1) \otimes \cdots \otimes v_k(m_k)$. In particular, for each $x \in \mathrm{X}_j$ let $\boldsymbol{v}(x) = 1_1 \otimes \cdots \otimes v_j(x) \otimes \cdots \otimes 1_k$ where 1_j is the multiplicative identity of A_j. We can now extend $\boldsymbol{v}$ by linearity to all polynomials in the domain $\mathbb{F}\langle \mathrm{X}_1\rangle \otimes \cdots \otimes \mathbb{F}\langle \mathrm{X}_k\rangle$.

Next, we define a *partial evaluation* of $f \in \mathbb{F}\langle \mathrm{X}_1\rangle \otimes \cdots \otimes \mathbb{F}\langle \mathrm{X}_k\rangle$ in $\boldsymbol{A}$. Let $k' < k$ and $\hat{\boldsymbol{A}} = (A_1, \ldots, A_{k'})$ be a k'-tuple of $\mathbb{F}$-algebras. A partial evaluation of $\mathbb{F}\langle \mathrm{X}_1\rangle \otimes \cdots \otimes \mathbb{F}\langle \mathrm{X}_k\rangle$ in $\hat{\boldsymbol{A}}$ is given by a k'-tuple of maps $\hat{\boldsymbol{v}} = (v_1, \ldots, v_{k'})$, where $v_i : \mathrm{X}_i \to A_i$. Now, we can define $\hat{\boldsymbol{v}} : \mathbb{F}\langle \mathrm{X}_1\rangle \otimes \cdots \otimes \mathbb{F}\langle \mathrm{X}_k\rangle \to A_1 \otimes \cdots \otimes A_{k'} \otimes \mathbb{F}\langle \mathrm{X}_{k'+1}\rangle \otimes \cdots \otimes \mathbb{F}\langle \mathrm{X}_k\rangle$ as follows. For a monomial $m = (m_1 \otimes \cdots \otimes m_k)$, $m_i \in \mathrm{X}_i^*$, we let $\hat{\boldsymbol{v}}(m) = v_1(m_1) \otimes \cdots \otimes v_{k'}(m_{k'}) \otimes m_{k'+1} \otimes \cdots \otimes m_k$. By linearity, the partial evaluation $\hat{\boldsymbol{v}}$ is defined for any $f \in \mathbb{F}\langle \mathrm{X}_1\rangle \otimes \cdots \otimes \mathbb{F}\langle \mathrm{X}_k\rangle$ where $\hat{\boldsymbol{v}}$ takes values in $A_1 \otimes \cdots \otimes A_{k'} \otimes \mathbb{F}\langle \mathrm{X}_{k'+1}\rangle \otimes \cdots \otimes \mathbb{F}\langle \mathrm{X}_k\rangle$.

Although it is implicit, we formally recall that when we consider ABPs over $\mathbb{F}\langle \mathrm{X}_1\rangle \otimes \cdots \otimes \mathbb{F}\langle \mathrm{X}_k\rangle$ the linear forms are defined over tensors of the form $1 \otimes \cdots \otimes x_{ij} \otimes \cdots \otimes 1$. These tensors play the role of an individual variable in the tensor product structure.

Some more notation. Let $S_{k,n,d,s}$ denote the set of all polynomials in $\mathbb{F}\langle \mathrm{X}_1\rangle \otimes \cdots \otimes \mathbb{F}\langle \mathrm{X}_k\rangle$ computed by ABPs of size s and layers $0, 1, \ldots, d$, and $n = |\mathrm{X}_i|$ for each $1 \le i \le k$. Following the notation in Theorem 2.1, we will denote by $\mathcal{H}_{k,n,d,s}$ the hitting set that we will construct for $S_{k,n,d,s}$. That is, $\mathcal{H}_{k,n,d,s}$ is a collection of evaluations in the ring of square matrices $\boldsymbol{v} = (v_1, \ldots, v_k)$, such that for any nonzero polynomial $f \in S_{k,n,d,s}$ there is an evaluation $\boldsymbol{v} = (v_1, \ldots, v_k) \in \mathcal{H}_{k,n,d,s}$ such that $\boldsymbol{v}(f)$ is a nonzero matrix. Recall from Theorem 2.1 that a quasi-polynomial size hitting set $\mathcal{H}_{1,n,d,s}$ for $S_{1,n,d,s}$ can be explicitly constructed. In the next lemma we describe an efficient bootstrapped construction of the hitting set $\mathcal{H}_{k,n,d,s}$ for the set $S_{k,n,d,s}$ of polynomials, from the hitting set $\mathcal{H}_{1,n,d,s}$.

▶ **Lemma 4.1.** *There is a set of evaluation maps $\mathcal{H}_{k,n,d,s} = \{(v_1, \ldots, v_k) : v_i \in \mathcal{H}_{1,n,d,s_k}\}$ where $s_k = s(d+1)^{(k-1)}$ such that, for $i \in [k]$, we have $v_i : \mathrm{X}_i \to \mathcal{M}_{d+1}(\mathbb{F})$, and $\mathcal{H}_{k,n,d,s}$ is a hitting set for the class of polynomials $S_{k,n,d,s}$. Moreover, the size of the set is at most $(nskd)^{O(k^2 \log d)}$, and it can be constructed in deterministic $(nskd)^{O(k^2 \log d)}$ time.*

The above lemma yields the identity test: we only need to evaluate the input polynomial on each point of the hitting set and check whether it is nonzero.

Before presenting the proof, we discuss two important ingredients. A polynomial f in $\mathbb{F}\langle \mathrm{X}_1\rangle \otimes \cdots \otimes \mathbb{F}\langle \mathrm{X}_k\rangle$ can be written as $f = \sum_{m \in X_k^*} f_m \otimes m$ where each m is a monomial over variables X_k and $f_m \in \mathbb{F}\langle \mathrm{X}_1\rangle \otimes \cdots \otimes \mathbb{F}\langle \mathrm{X}_{k-1}\rangle$. Given that f has a small ABP, we first show that each polynomial f_m also has a small ABP.

▶ **Lemma 4.2.** *For each $f \in S_{k,n,d,s}$ and $m \in \mathrm{X}_k^*$, the polynomial $f_m \in \mathbb{F}\langle \mathrm{X}_1\rangle \otimes \cdots \otimes \mathbb{F}\langle \mathrm{X}_{k-1}\rangle$ has an ABP of size $s(d+1)$ and $d+1$ layers.*

Proof. Suppose $f \in \mathbb{F}\langle \mathrm{X}_1\rangle \otimes \cdots \otimes \mathbb{F}\langle \mathrm{X}_k\rangle$ has an ABP B of size s and d layers, and the monomial $m = x_{i_1 k} x_{i_2 k} \cdots x_{i_\ell k}$ where some of the indices could be repeated. We will construct an ABP of size $s(d+1)$ for the polynomial f_m. First, we identify each variable $1 \otimes \cdots \otimes x_{ij} \otimes \cdots \otimes 1$ as x_{ij} and construct the following ABP B' from B:

For every node u in the ABP B, we have nodes $(u, i), 0 \le i \le \ell$ in the ABP B'. We now describe the edges of B' and the edge labels. In the ABP B, let (u, v) be an edge, where u is in layer j and v is in layer $j+1$, for some $j \le d-1$. We can write the linear form labeling (u, v) as a sum $L_1 + L_2$, where L_1 is an affine linear form in variables from $X \setminus X_k$, and L_2 is a homogeneous linear form in variables from X_k.

For $0 \le r \le \ell - 1$: we put an edge from (u, r) to (v, r) with label L_1. For $0 \le r \le \ell - 1$: we put an edge from (u, r) to $(v, r+1)$ with edge label $\alpha \cdot x_{i_{r+1} k}$ if the coefficient of $x_{i_{r+1} k}$ in L_2 is $\alpha \ne 0$. If s and t are the source and sink nodes of the ABP B, we designate $(s, 0)$ and (t, ℓ) as the source and sink nodes of the ABP B'.

It is evident from the construction that the ABP B' has at most $s(d+1)$ many nodes. Furthermore, the only nonzero monomials in the polynomial computed by B' are of the form $m' \otimes m$, where m' is a monomial over the letters $X \setminus X_k$, and the coefficient of $m' \otimes m$ is the same as its coefficient in polynomial f. It follows, that B' computes the polynomial $f_m \otimes m$, and we can obtain an ABP for f_m by setting to 1 all the variables occurring in m. This completes the proof. ◀

For a polynomial f in $\mathbb{F}\langle \mathrm{X}_1\rangle \otimes \cdots \otimes \mathbb{F}\langle \mathrm{X}_k\rangle$, consider a partial evaluation $\boldsymbol{v} = (v_1, \ldots, v_{k-1})$ such that each $v_i : \mathrm{X}_i \to \mathcal{M}_{t_i}(\mathbb{F})$. The evaluation $\boldsymbol{v}(f)$ is a $T \times T$ matrix with entries from $\mathbb{F}\langle \mathrm{X}_k\rangle$, where $T = t_1 t_2 \cdots t_{k-1}$.

▶ **Lemma 4.3.** *For each $p, q \in [T]$ and $f \in S_{k,n,d,s}$, the $(p,q)^{th}$ entry of $\boldsymbol{v}(f)$ can be computed by an ABP of size sT and $d+1$ layers.*

The proof is routine and given in the appendix. Now we are ready to prove Lemma 4.1.

Proof of Lemma 4.1. The proof is by induction on k. For the base case $k = 1$ the hitting set $\mathcal{H}_{1,n,d,s}$ of Theorem 2.1 suffices. We can write each nonzero $f \in S_{k,n,d,s}$ as $f = \sum_{m \in X_k^*} f_m \otimes m$, where $m \in \mathrm{X}_k^*$ and $f_m \in \mathbb{F}\langle \mathrm{X}_1 \rangle \otimes \cdots \otimes \mathbb{F}\langle \mathrm{X}_{k-1} \rangle$. Since $f \not\equiv 0$ we have $f_m \not\equiv 0$ for some $m \in \mathrm{X}_k^*$. By Lemma 4.2, for each $m \in {\mathrm{X}_k}^*$ the polynomial $f_m \in \mathbb{F}\langle \mathrm{X}_1 \rangle \otimes \cdots \otimes \mathbb{F}\langle \mathrm{X}_{k-1} \rangle$ has an ABP of size $s(d+1)$. Let $s' = s(d+1)$.

By induction hypothesis, f_m is nonzero on some point in the set : $\mathcal{H}_{k-1,n,d,s'} = \{(v_1, v_2, \dots, v_{k-1}) | v_i \in \mathcal{H}_{1,n,d,s'_{k-1}}\}$ where $s'_{k-1} = s'(d+1)^{k-2} = s(d+1)^{k-1}$. Hence, there is an evaluation $\boldsymbol{v'} \in \mathcal{H}_{k-1,n,d,s'}$ such that $\boldsymbol{v'}(f_m)$ is a nonzero matrix of dimension $(d+1)^{k-1}$. Interpreting $\boldsymbol{v'}$ as a *partial evaluation* for f, we observe that $\boldsymbol{v'}(f)$ is a $(d+1)^{k-1} \times (d+1)^{k-1}$ matrix with entries from $\mathbb{F}\langle \mathrm{X}_k \rangle$. Since $\boldsymbol{v'}(f_m) \neq 0$, it follows that some $(p,q)^{th}$ entry of $\boldsymbol{v'}(f)$ is a nonzero polynomial $g \in \mathbb{F}\langle \mathrm{X}_k \rangle$. By Lemma 4.3, each entry of $\boldsymbol{v'}(f)$ has an ABP of size $s(d+1)^{k-1}$. In particular, $g \in S_{1,n,d,s(d+1)^{k-1}}$ and it follows from Theorem 2.1 that there is a an evaluation v'' in $\mathcal{H}_{1,n,d,s(d+1)^{k-1}}$ such that $v''(g)$ is a nonzero matrix of dimension $(d+1) \times (d+1)$.

Thus, for the combined evaluation map $\boldsymbol{v} = (\boldsymbol{v'}, v'')$, $\boldsymbol{v}(f)$ is a nonzero matrix of dimension $(d+1)^k \times (d+1)^k$. Define $\mathcal{H}_{k,n,d,s} = \{(v_1, \dots, v_k) : v_i \in \mathcal{H}_{1,n,d,s_k}\}$, where $s_k = s(d+1)^{k-1}$. However, by induction hypothesis, we have $\boldsymbol{v'} = (v_1, \dots, v_{k-1}) \in \mathcal{H}_{k-1,n,d,s(d+1)}$ where each $v_i \in \mathcal{H}_{1,n,d,s(d+1)^{k-1}}$. Therefore, $\boldsymbol{v} = (\boldsymbol{v'}, v'') \in \mathcal{H}_{k,n,d,s}$ and $\mathcal{H}_{k,n,d,s}$ is a hitting set for the class of polynomials $S_{k,n,d,s}$.

Finally, note that $|\mathcal{H}_{k,n,d,s}| = |\mathcal{H}_{1,n,d,s_k}|^k$. Since $|\mathcal{H}_{1,n,d,s_k}| \leq (nds_k)^{O(\log d)}$, it follows that $|\mathcal{H}_{k,n,d,s}| \leq (nskd)^{O(k^2 \log d)}$, and the hitting set $\mathcal{H}_{k,n,d,s}$ can be constructed in the claimed running time[10]. For zero testing, we need to evaluate the input ABP on the matrices of the hitting set, and this can be done in time polynomial in the input size and the size of the hitting set by matrix additions and multiplications. ◀

5 Randomized Zero Testing of Weighted Automata Over k-Monoids

We now consider pc monoids more general than k-clique monoids. A *k-monoid* is a pc monoid M whose non-commutation graph G_M has a 2-covering $\{G_1, G_2\}$ such that G_1 has an edge clique cover of size k' and G_2 has a vertex cover of size $k - k'$, for some k'. It follows that G_M has a k-covering of cliques and star graphs. We assume that this k-covering of G_M is given as part of the input. Let $\mathbb{F}\langle M \rangle$ denote the $\mathbb{F}$-algebra generated by the monoid M.

▶ **Lemma 5.1.** *Let $\{M_i\}_{i=1}^k$ be pc monoids defined over disjoint variable sets $\{\mathrm{X}_i\}_{i=1}^k$, respectively. For each i, suppose $\mathcal{A}_i$ is a randomized procedure that outputs an evaluation $v_i : \mathbb{F}\langle M_i \rangle \to \mathcal{M}_{t_i(d)}(\mathbb{F})$ such that for any polynomial g_i in $\mathbb{F}\langle M_i \rangle$ of degree at most d, g_i is nonzero if and only if $v_i(g_i)$ is a nonzero matrix with probability at least $1 - \frac{1}{2k}$.*

Then, for the evaluation $\boldsymbol{v} : \mathbb{F}\langle M_1 \rangle \otimes \cdots \otimes \mathbb{F}\langle M_k \rangle \to \mathcal{M}_{t_1(d)}(\mathbb{F}) \otimes \cdots \otimes \mathcal{M}_{t_k(d)}(\mathbb{F})$ such that $\boldsymbol{v} = (v_1, \dots, v_k)$ and any nonzero polynomial $f \in \mathbb{F}\langle M_1 \rangle \otimes \cdots \otimes \mathbb{F}\langle M_k \rangle$ of degree at most d, the matrix $\boldsymbol{v}(f)$ is nonzero with probability at least $1/2$.

[10] Independent to the context of the current paper, bootstrapping hitting sets has found other interesting applications in arithmetic circuit complexity [1].

Proof. The proof is by induction on k. For the base case $k = 1$, it is trivial. Let us fix an $f \in \mathbb{F}\langle M_1\rangle \otimes \cdots \otimes \mathbb{F}\langle M_k\rangle$ of degree at most d such that $f \not\equiv 0$. The polynomial f can be written as $f = \sum_{m\in M_k} f_m \otimes m$ where m are the words over the pc monoid M_k and $f_m \in \mathbb{F}\langle M_1\rangle \otimes \cdots \otimes \mathbb{F}\langle M_{k-1}\rangle$. Since $f \not\equiv 0$ we must have $f_m \not\equiv 0$ for some $m \in M_k$.

Now, inductively we have the evaluation $\boldsymbol{v'} = (v_1, \ldots, v_{k-1})$ for the class of polynomials in $\mathbb{F}\langle M_1\rangle \otimes \cdots \otimes \mathbb{F}\langle M_{k-1}\rangle$ of degree at most d. Since $f_m \not\equiv 0$, with high probability $\boldsymbol{v'}(f_m)$ is a nonzero matrix of dimension $\prod_{i=1}^{k-1} t_i(d)$. By induction the failure probability is bounded by $\frac{k-1}{2k}$.

As $\boldsymbol{v'}$ is a *partial evaluation* for f, we observe that $\boldsymbol{v'}(f)$ is a matrix of dimension $\prod_{i=1}^{k-1} t_i(d)$ whose entries are polynomials in $\mathbb{F}\langle M_k\rangle$. Since $\boldsymbol{v'}(f_m) \neq 0$ we conclude that some $(p,q)^{th}$ entry of $\boldsymbol{v'}(f)$ contains a nonzero polynomial $g \in \mathbb{F}\langle M_k\rangle$ of degree at most d. Choose the evaluation $v_k \in S_k$ which is the output of the randomized procedure $\mathcal{A}_k$, such that $v_k(g)$ is a nonzero matrix of dimension $t_k(d)$. Hence, for the combined evaluation $\boldsymbol{v} = (\boldsymbol{v'}, v_k)$, $\boldsymbol{v}(f)$ is a nonzero matrix of dimension $\prod_{i=1}^{k} t_i(d)$. A union bound shows that the failure probability is at most $1/2$. ◀

For the proof of Theorem 1.4, we first give a randomized polynomial-time identity testing algorithm for polynomials over pc monoids whose non-commutation graph is a star graph.

▶ **Lemma 5.2.** *Let $M = ((\mathrm{X} \cup y)^*, I)$ be a monoid whose non-commutation graph G_M is a star graph with center y. Then for any constant k, there is a randomized procedure that outputs an evaluation $\boldsymbol{v} : \mathrm{X} \cup \{y\} \to \mathrm{Mat}_{t(d)}(\mathbb{F})$ where $t(d)$ is at most d, such that for any polynomial $f \in \mathbb{F}\langle M\rangle$ of degree at most d, the polynomial f is nonzero if and only if $\boldsymbol{v}(f)$ is a nonzero matrix. The success probability of the algorithm is at least $1 - \frac{1}{2k}$.*

Proof. If f is nonzero, then there exists a monomial m in M with nonzero coefficient. The idea is to isolate all monomials in $\{\mathrm{X} \cup y\}^*$ that are equivalent to m in M. Let the degree of y in monomial m be $\ell \le d$. Then m can be written as $m = m_1 y m_2 \cdots m_\ell y m_{\ell+1}$ where each m_i is a word in X^*. As X is a commuting set of variables, any permutation of m_i produces a monomial equivalent to m in M. Now consider the automaton in Figure 1.

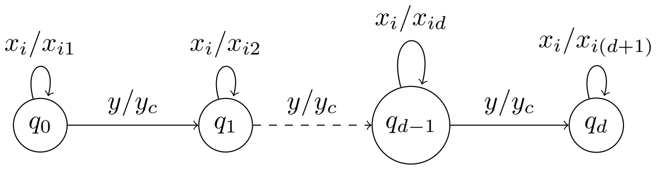

Figure 1 The transition diagram of the automaton.

Let m as $m = m_1 y m_2 \cdots m_\ell y m_{\ell+1}$, where each m_i is a maximal substring of m in X^*. We refer to the m_i as blocks. The above automaton keeps count of blocks as it scans the monomial m. As it scans m, if the automaton is in the j^{th} block, it substitutes each variable $x_i \in \mathrm{X}$ read by a corresponding commuting variable x_{ij} where the index j encodes the block number. The y variable is renamed by a commutative variable y_c. The transition matrices N_{x_i} and N_y of dimension $d+1$. The transition matrices are explicitly given below.

$$N_{x_i} = \begin{bmatrix} x_{i1} & 0 & 0 & \ldots & 0 \\ 0 & x_{i2} & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \ldots & x_{id} & 0 \\ 0 & 0 & \ldots & 0 & x_{i(d+1)} \end{bmatrix}, \qquad N_y = \begin{bmatrix} 0 & y_c & 0 & \ldots & 0 \\ 0 & 0 & y_c & \ldots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 & y_c \\ 0 & 0 & \ldots & 0 & 0 \end{bmatrix}.$$

Now we explain this matrix substitution. Let $f = \sum_m \alpha_m m$, where $\alpha_m \in \mathbb{F}$. We write $f = \sum_{\ell=1}^{d} f_\ell$, where $f_\ell = \sum_{m:\deg_y(m)=\ell} \alpha_m m$. That is, f_ℓ is the part of f consisting of monomials m with y-degree $\deg_y(m) = \ell$.

From the description of the automaton, we can see that for each $\ell \in [d]$, the $(0, \ell)^{th}$ entry of the output matrix is the commutative polynomial $f_\ell^c \in \mathbb{F}[\{x_{i,j}\}_{1\le i\le n, 1\le j\le d+1}, y_c]$. The construction ensures the following: For each $0 \le \ell \le d$, $f_\ell = 0$ if and only if $f_\ell^c = 0$.

The randomized identity test is by substituting random scalar values for the commuting variables x_{ij} and y_c from a set $S \subseteq \mathbb{F}$ of size at least $2kd$, such that the output matrix becomes nonzero. The bound on the success probability follows from Polynomial Identity Lemma [25, 20, 7]. ◀

Now we are ready to prove Theorem 1.4.

Proof of Theorem 1.4. Let M' be a pc monoid whose non-commutation graph $G_{M'}$ is a clique. Let $g \in \mathbb{F}\langle M' \rangle$ be a nonzero polynomial of degree at most d. By the Amitsur-Levitzki Theorem [2], if we substitute variables $x_i \in M'$ by generic matrix of size d over the variables $\{x_{u,v}^{(i)}\}_{1\le u,v\le d}$, the output matrix is nonzero [11]. Moreover, the entries of the output matrix are commutative polynomials of degree at most d in the variables $\{x_{u,v}^{(i)}\}_{1\le i\le n, 1\le u,v\le d}$. It suffices to randomly substitute for each $x_{u,v}^{(i)}$ variable from a set $S \subseteq \mathbb{F}$ of size at least $2kd$. This defines the evaluation map $v : \mathbb{F}\langle M' \rangle \to \mathbb{M}_d(\mathbb{F})$. The resulting identity test succeeds with probability at least $1 - \frac{1}{2k}$. For the star graphs, the evaluation map is already defined in Lemma 5.2.

Given a $\mathbb{F}$-weighted automaton A of size s over a k-monoid $M = (\mathrm{X}^*, I)$, by Theorem 3.5, the zero testing of A reduces to identity testing of a collection of ABPs of the form : $f = \lambda^T N^d \mu$ over $\mathbb{F}\langle M \rangle$, where N is the transition matrix of A and d is bounded by $O(ns^2k)$. Now, to test identity of f where M is a k-monoid, it suffices to test identity of $\psi(f)$ where ψ is the injective homomorphism from Lemma 3.1. Now $\psi(f)$ in $\mathbb{F}\langle M_1' \rangle \otimes \cdots \otimes \mathbb{F}\langle M_k' \rangle$, where for each $i \in [k]$ the non-commutation graph of M_i' is either a clique or a star.

By Lemma 5.1, we can construct the evaluation map $\boldsymbol{v} = v_1 \otimes v_2 \otimes \cdots \otimes v_k$ where for each $i \in [k]$, v_i is an evaluation map for either a clique or a star graph depending on M_i'. The range of $\boldsymbol{v}$ is matrices of dimension at most d^k, which is bounded by $(sn)^{O(k)}$ as d is bounded by $O(ns^2k)$. This completes the proof of Theorem 1.4. ◀

Concluding Remarks

The bootstrapped construction presented in Section 4 designs a quasi-polynomial time algorithm for the k-clique monoid problem which uses evaluation over a suitable matrix algebra. However, to design a polynomial-time algorithm, one may try to exploit finer structures in the problem than evaluating it over a matrix algebra. The obvious natural idea is to generalize the white-box polynomial-time identity testing algorithm for noncommutative ABPs [18] in the pc monoid setting. However, it is unclear whether such a generalization is possible.

[11] In fact the Amitsur-Levitzki theorem guarantees that generic matrices of size $\lceil \frac{d}{2} \rceil + 1$ suffice [2].

References

1 Manindra Agrawal, Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Hitting-sets for ROABP and sum of set-multilinear circuits. *SIAM J. Comput.*, 44(3):669–697, 2015. doi:10.1137/140975103.

2 A. S. Amitsur and J. Levitzki. Minimal identities for algebras. *Proceedings of the American Mathematical Society*, 1(4):449–463, 1950. URL: http://www.jstor.org/stable/2032312.

3 C. Beeri. An improvement on Valiant's decision procedure for equivalence of deterministic finite turn pushdown machines. *Theoretical Computer Science*, 3(3):305–320, 1976. doi:10.1016/0304-3975(76)90049-9.

4 J. Berstel and C. Reutenauer. *Noncommutative Rational Series with Applications*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2011.

5 Malcolm Bird. The equivalence problem for deterministic two-tape automata. *J. Comput. Syst. Sci.*, 7(2):218–236, 1973. doi:10.1016/S0022-0000(73)80045-5.

6 Mireille Clerbout, Michel Latteux, and Yves Roos. Decomposition of partial commutations. In Mike Paterson, editor, *Automata, Languages and Programming, 17th International Colloquium, ICALP90, Warwick University, England, UK, July 16-20, 1990, Proceedings*, volume 443 of *Lecture Notes in Computer Science*, pages 501–511. Springer, 1990. doi:10.1007/BFb0032054.

7 Richard A. Demillo and Richard J. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7(4):193–195, 1978. doi:10.1016/0020-0190(78)90067-4.

8 Volker Diekert. *Combinatorics on Traces*, volume 454 of *Lecture Notes in Computer Science*. Springer, 1990. doi:10.1007/3-540-53031-2.

9 Volker Diekert, Markus Lohrey, and Alexander Miller. Partially commutative inverse monoids. In Rastislav Kralovic and Pawel Urzyczyn, editors, *Mathematical Foundations of Computer Science 2006, 31st International Symposium, MFCS 2006, Stará Lesná, Slovakia, August 28-September 1, 2006, Proceedings*, volume 4162 of *Lecture Notes in Computer Science*, pages 292–304. Springer, 2006. doi:10.1007/11821069_26.

10 Manfred Droste and Paul Gastin. On recognizable and rational formal power series in partially commuting variables. In Pierpaolo Degano, Roberto Gorrieri, and Alberto Marchetti-Spaccamela, editors, *Automata, Languages and Programming, 24th International Colloquium, ICALP'97, Bologna, Italy, 7-11 July 1997, Proceedings*, volume 1256 of *Lecture Notes in Computer Science*, pages 682–692. Springer, 1997. doi:10.1007/3-540-63165-8_222.

11 Samuel Eilenberg. *Automata, Languages, and Machines (Vol A)*. Pure and Applied Mathematics. Academic Press, 1974.

12 Michael A. Forbes and Amir Shpilka. Quasipolynomial-time identity testing of non-commutative and read-once oblivious algebraic branching programs. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 243–252, 2013. doi:10.1109/FOCS.2013.34.

13 Emily P. Friedman and Sheila A. Greibach. A polynomial time algorithm for deciding the equivalence problem for 2-tape deterministic finite state acceptors. *SIAM J. Comput.*, 11:166–183, 1982.

14 T. V. Griffiths. The unsolvability of the equivalence problem for nondeterministic generalized machines. *J. ACM*, 15(3):409–413, 1968. doi:10.1145/321466.321473.

15 Tero Harju and Juhani Karhumäki. The equivalence problem of multitape finite automata. *Theor. Comput. Sci.*, 78(2):347–355, 1991. doi:10.1016/0304-3975(91)90356-7.

16 Antoni W. Mazurkiewicz. Trace theory. In *Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986, Part II, Proceedings of an Advanced Course, Bad Honnef, Germany, 8-19 September 1986*, pages 279–324, 1986. doi:10.1007/3-540-17906-2_30.

17 Noam Nisan. Lower bounds for non-commutative computation (extended abstract). In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 410–418, 1991. doi:10.1145/103418.103462.

18 Ran Raz and Amir Shpilka. Deterministic polynomial identity testing in non-commutative models. *Computational Complexity*, 14(1):1–19, 2005. doi:10.1007/s00037-005-0188-8.

19 M.P. Schützenberger. On the definition of a family of automata. *Information and Control*, 4(2):245–270, 1961. doi:10.1016/S0019-9958(61)80020-X.

20 Jacob T. Schwartz. Fast probabilistic algorithm for verification of polynomial identities. *J. ACM.*, 27(4):701–717, 1980.

21 W. Tzeng. A polynomial-time algorithm for the equivalence of probabilistic automata. *SIAM Journal on Computing*, 21(2):216–227, 1992.

22 Leslie G. Valiant. The equivalence problem for deterministic finite-turn pushdown automata. *Information and Control*, 25(2):123–133, 1974. doi:10.1016/S0019-9958(74)90839-0.

23 Stefano Varricchio. On the decidability of equivalence problem for partially commutative rational power series. *Theor. Comput. Sci.*, 99(2):291–299, 1992. doi:10.1016/0304-3975(92)90354-I.

24 James Worrell. Revisiting the equivalence problem for finite multitape automata. In *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II*, pages 422–433, 2013. doi:10.1007/978-3-642-39212-2_38.

25 R. Zippel. Probabilistic algorithms for sparse polynomials. In *Proc. of the Int. Sym. on Symbolic and Algebraic Computation*, pages 216–226, 1979.

A The Proof of Lemma 3.1

Proof. It is straightforward to check that ψ is a ring homomorphism. To show the injectivity, it is enough to show that $\psi(m) = \psi(m')$ implies $m = m'$ in M for any words $m, m' \in M$. We prove the claim by induction on the length of words in M. Suppose that for words $m \in M$ of length at most ℓ, if m' is not $\tilde{I}$-equivalent to m then $\psi(m) \neq \psi(m')$. The base case, for $\ell = 0$ clearly holds.

Now, suppose $m = x \cdot m_1 \in \mathrm{X}^{\ell+1}$ for $x \in \mathrm{X}$ and $\psi(m) = \psi(m')$.

▷ Claim A.1. For some $m_2 \in M$, $m' = x \cdot m_2$ in M.

Proof. Assume, to the contrary, that there is no $m_2 \in M$ such that $m' = x \cdot m_2$. Let $J = \{j \in [k] \mid x \in \mathrm{X}_j\}$. If the variable x does not occur in m' then $m|_{\mathrm{X}_j} \neq m'|_{\mathrm{X}_j}$ for each $j \in J$. This implies that $\psi(m) \neq \psi(m')$ which is a contradiction.

On other hand, suppose x occurs in m' and it cannot be moved to the leftmost position in m' applying the commutation relations in I. Then we must have $m' = ayxb$ for some $y \in \mathrm{X}_j$ and $j \in J$, where $a, b \in \mathrm{X}^*$, for the leftmost occurrence of x in m'. Hence $m|_{\mathrm{X}_j} \neq m'|_{\mathrm{X}_j}$, because x is the first variable in $m|_{\mathrm{X}_j}$ and x comes after y in $m'|_{\mathrm{X}_j}$. Therefore, $\psi(m) \neq \psi(m')$ which is a contradiction. ◁

Now, $\psi(x \cdot m_1) = \psi(x \cdot m_2)$ implies that $\psi(m_1) = \psi(m_2)$. Both m_1 and m_2 are of length ℓ. By induction hypothesis it follows that $m_1 = m_2$, and hence $m = m'$. ◀

B The Proof of Lemma 4.3

Proof. In effect the edges of the input branching program B are now labelled by matrices of dimension T with entries are linear forms over the variables X'_k. To show that each entry of the final $T \times T$ matrix can be computed by an ABP of size sT, let us fix some (i, j) such that $1 \leq i, j \leq T$ and construct an ABP B'_{ij} computing the polynomial in the $(i, j)^{th}$ entry.

The construction of B'_{ij} is as follows. We make T copies of each node p (except the source and sink node) of B and label it as (p, k) for each $k \in [T]$. Let us fix two nodes p and q from B such that there is a $T \times T$ matrix M_{pq} labelling the edge (p, q) after the substitution. Then, for each $j_1, j_2 \in [T]$, add an edge between (p, j_1) and (q, j_2) in B'_{ij} and label it by the $(j_1, j_2)^{th}$ entry of M_{pq}. When p is the source node, for each $j_2 \in T$, add an edge between the source node and (q, j_2) in B'_{ij} and label it by the $(i, j_2)^{th}$ entry of M_{pq}. Similarly, when q is the sink node, for each $j_1 \in T$, add an edge between (p, j_1) and the sink node in B'_{ij} and label it by the $(j_1, j)^{th}$ entry of M_{pq}.

We just need to argue that the intermediate edge connections simulate matrix multiplications correctly. This is simple to observe, since for each path

$$\mathcal{P} = \{(s, p_1), (p_1, p_2), \ldots, (p_{\ell-1}, t)\}$$

in B (where s, t are the source and sink nodes respectively) and each $(j_1, \ldots, j_{\ell-1})$ such that $1 \leq j_1, \ldots, j_{\ell-1} \leq T$, there is a path $(s, (p_1, j_1)), ((p_1, j_1), (p_2, j_2)), \ldots, ((p_{\ell-1}, j_{\ell-1}), t)$ in B'_{ij} that computes $M_{(s,p_1)}[i, j_1]M_{(p_1,p_2)}[j_1, j_2] \cdots M_{p_{\ell-1},t}[j_{\ell-1}, j]$ where $M_{(p,q)}$ is the $T \times T$ matrix labelling the edge (p, q) in B. The size of B'_{ij} is sT, and the number of layers is $d + 1$. ◀