Multiplicity Equivalence Testing of Automata over Partially Commutative Monoids

V. Arvind
Institute of Mathematical Sciences (HBNI), Chennai, India
email: arvind@imsc.res.in

Abhranil Chatterjee
Institute of Mathematical Sciences (HBNI), Chennai, India
email: abhranilc@imsc.res.in

Rajit Datta
Chennai Mathematical Institute, Chennai, India
email: rajit@cmi.ac.in

Partha Mukhopadhyay
Chennai Mathematical Institute, Chennai, India
email: partham@cmi.ac.in

Abstract

We study *multiplicity equivalence* testing of automata over partially commutative monoids (pc monoids) and show efficient algorithms in special cases, exploiting the structure of the underlying non-commutation graph of the monoid.

Specifically, if the clique cover number of the non-commutation graph (the minimum number of cliques covering the graph) of the pc monoid is a constant, we obtain a deterministic quasi-polynomial time algorithm. As a consequence, we also obtain the first deterministic quasi-polynomial time algorithms for multiplicity equivalence testing of k-tape automata and for equivalence testing of deterministic k-tape automata for constant k. Prior to this, a randomized polynomial-time algorithm for the above problems was shown by Worrell [20].

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

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

Keywords and phrases Automata Equivalence, Partially Commutative Monoid, Schützenberger's Theorem

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

1 Introduction

Testing equivalence of multi-tape finite automata is a fundamental problem in automata theory. For a k-tape automaton, we usually denote by $\Sigma_1, \dots, \Sigma_k$ the mutually disjoint alphabets for the k tapes, and 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. It is well-known that equivalence testing of multi-tape *non-deterministic* automata is undecidable [10].

For 2-tape *deterministic* automata equivalence testing was shown to be decidable in the 1970's [4, 19]. In [2] an exponential upper bound was shown. Subsequently, a polynomial-time algorithm was obtained [9] 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 multi-tape weighted automata. Intuitively, the multiplicity equivalence testing problem is to decide whether for each tuple in the product monoid $\Sigma_1^* \times \cdots \times \Sigma_k^*$, the numbers of accepting paths in the two input automata are the same. Since a deterministic automaton has at most one accepting path for each word, equivalence testing of two deterministic k-tape automata coincides with multiplicity equivalence testing. More generally, for weighted automata, multiplicity equivalence testing is to decide if the coefficient of each word (over a field or ring) is the same in the given automata. Multiplicity equivalence testing is in deterministic polynomial time for one-tape automata [16, 18]. Such an algorithm for the k-tape case remained elusive for a long time. Multiplicity equivalence testing of k-tape non-deterministic automata was shown *decidable* by Harju and Karhumäki [11] using the theory of free groups. No nice complexity-theoretic upper bound was known, until recently Worrell [20] obtained a *randomized* polynomial-time algorithm for testing the multiplicity equivalence of k-tape non-deterministic automata (and equivalence testing of deterministic k-tape automata) for any constant k. Worrell takes a different approach via Polynomial Identity Testing (PIT). In [20], Worrell explicitly raised the problem of finding an efficient *deterministic* algorithm for multiplicity equivalence problem for k-tape automata for any fixed k.

In this paper, we show that the multiplicity equivalence testing for k-tape automata can be solved in *deterministic* quasi-polynomial time. This immediately yields the first deterministic quasi-polynomial time algorithm to check the equivalence of deterministic k-tape automata, making progress on a question asked earlier [9, 11]. In fact, our proof technique shows a stronger result that we explain now. 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 whereas the 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-commutative relations: (x, y) is an edge if and only if x and y do not commute. For the k-tape case, the vertex set of G_M is $\Sigma_1 \cup \ldots \cup \Sigma_k$ and G_M is clearly the union of k disjoint cliques, induced by each Σ_i, forming a clique cover of size k. In this paper, we obtain a multiplicity equivalence testing algorithm over any pc monoid whose non-commutation graph has a constant size clique cover (*not necessarily* disjoint). In short, we call such monoids as *k-clique monoids* where the clique cover size is bounded by k.

▶ **Theorem 1.** *Let A and B be $\mathbb{F}$-weighted automata of total size s over a pc monoid M for which the non-commutation graph G_M has a clique cover of size k. Then, the multiplicity equivalence of A and B can be decided in deterministic $(nks)^{O(k^2 \log ns)}$ time. Here n is the size of the alphabet of M, and the clique cover is given as part of the input.*

As an immediate corollary, it yields a deterministic quasi-polynomial time algorithm for multiplicity equivalence of k-tape automata (also equivalence testing of deterministic k-tape automata). Notice that, for the k-tape case, the clique cover of size k is also part of the input since for each $i \in [k]$, the i^{th} tape alphabet Σ_i is explicitly given.

Next we address multiplicity equivalence over more general partially commutative monoids M. M is a *k-monoid* if its non-commutation graph G_M is a union $G_1 \cup G_2$ of two graphs, where G_1 has a clique cover of size at most k' and G_2 has a vertex cover of size at most $k - k'$ (hence its edges can be covered by $k - k'$ many star graphs). We show that multiplicity equivalence testing over k-monoids has a randomized polynomial-time algorithm. This generalizes Worrell's result [20].

▶ **Theorem 2.** *Given as input two $\mathbb{F}$-weighted automata of size s over a k-monoid M, their multiplicity equivalence can be decided in randomized $(ns)^{O(k)}$ time. Here n is the size of the alphabet of M, and the covering for G_M by at most k cliques and star graphs is also given as part of the input.*

▶ Remark 3. What is the complexity of multiplicity equivalence testing over general pc monoids? The non-commutation graph G_M of any pc monoid $M = (\mathrm{X}^*, I)$ has a clique covering of size bounded by $\binom{|\mathrm{X}|}{2}$. Hence, the above results give an exponential-time algorithm. Note that if G_M has an *induced matching* 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 11 shown in Section 3, that multiplicity equivalence testing over arbitrary pc monoids is deterministic polynomial-time reducible to multiplicity equivalence testing over matching monoids (if G_M has isolated vertices, one can add a new vertex (variable) for each isolated vertex and introduce a matching edge between them).

Various automata-theoretic problems have been studied in the setting of pc monoids. For example, pc monoids have found applications in modelling the behaviour of concurrent systems [12]. Droste and Gastin [7] have studied the relation between recognizability and rationality over pc monoids. Broadly, it is interesting to understand and identify the results in algebraic automata theory that can be generalized to the setting of pc monoids.

Proof Overview : Now we briefly discuss the main ideas behind our results. Worrell's key insight [20] is to reduce k-tape automata equivalence problem to a suitable instance of polynomial identity testing over non-commuting variables, which can be solved in randomized polynomial time [1, 5, 13]. Our strategy too is to carry out reductions to polynomial identity testing problem. Since we are considering automata over general pc monoids and we aim to design efficient deterministic algorithms, we require additional ideas. First, we develop an algebraic framework to transfer multiplicity equivalence testing over general pc monoids to pc monoids whose non-commutation graphs are a disjoint union of cliques. This allows us to prove a Schützenberger [16] type theorem [1] over *general partially commutative monoids* which says that any nonzero weighted automata of size s over any pc monoid, must have a nonzero word within length $\mathrm{poly}(s, n)$ where n is the size of the alphabet. We apply this result to reduce multiplicity equivalence testing to polynomial identity testing for algebraic branching programs over pc monoids. It turns out that the latter problem can be solved by suitably adapting a black-box polynomial identity test for non-commutative algebraic branching programs based on hitting sets due to Forbes and Shpilka [8]. Our algorithm recursively builds on this result, ensuring that the resulting hitting set remains of quasi-polynomial size, like the Forbes-Shpilka hitting set [8]. This requires coupling a result of Schützenberger related to the Hadamard product of weighted automata ([15], Theorem 3.2)

[1] The theorem was originally proved over subrings of division rings.

with our algebraic framework. The proof of Theorem 2 also follows a similar line of argument. First we give a randomized polynomial-time identity testing algorithm over pc monoids whose non-commutation graph is a star graph. Then a composition lemma yields an identity testing algorithm over k-monoids.

The paper is organized as follows. In Section 2, we give some background. We prove a Schützenberger type theorem for automata over pc monoids in Section 3. Theorem 1 is presented in Section 4, and Theorem 2 in Section 5. Some proof details are in the appendix.

2 Preliminaries

We recall some basic definitions, mainly from automata theory and arithmetic circuit complexity and define some notations used in the subsequent sections.

Let $\mathbb{F}$ be a field. The results presented in the subsequent sections are independent of the characteristic of $\mathbb{F}$ and hold for fields of sufficiently large size. Let $\mathcal{M}_t(\mathbb{F})$ denote 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 product $A \otimes B$ is defined as $(a_{ij}B)_{1 \le i \le m, 1 \le j \le n}$. The dimension of $A \otimes B$ is $pm \times qn$. For a series (resp. polynomial) S and a word 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) $\mathrm{X} = \{x_1, \ldots, x_n\}$. We also consider coverings of a graphs, a graph $G = (\mathrm{X}, E)$ is said to have a graph covering $\{G_i = (X_i, E_i)\}_{i=1}^{k}$ of size k if $X = \bigcup_{i=1}^{k} X_i$ and $E = \bigcup_{i=1}^{k} E_i$.

Automata Theory : A $\mathbb{F}$-weighted automaton A of size s is a six-tuple $(Q, \mathrm{X}, E, \boldsymbol{u}, \boldsymbol{v}, wt)$ over X where Q is a set of s states, $\boldsymbol{u}, \boldsymbol{v} : Q \to \mathbb{F}$ are weight functions of initial and final states, and transitions $E \subseteq Q \times \mathrm{X} \times Q$ and $wt : Q \times \mathrm{X} \times Q \to \mathbb{F}$ is the weight assigned to each transition. It computes a series S in $\mathbb{F}\langle\langle \mathrm{X} \rangle\rangle$ where for every word $w \in \mathrm{X}^*$, $[w]S$ is the sum of the weights of w in each accepting path of w in A. The next (folklore) proposition shows that the ϵ-transitions can be removed from a weighted automaton. The proof of this proposition is in Section A of the appendix.

▶ **Proposition 4.** *Given a weighted automaton A there is a weighted automaton B of same size without ϵ-transitions that computes the same series as A.*

Henceforth, we consider all automata are free of ϵ-transitions. A series S computed by an $\mathbb{F}$-weighted automata of size s (also called a recognisable series) can be defined as a monoid homomorphism $\mu : \mathrm{X}^* \mapsto \mathcal{M}_s(\mathbb{F})$ defined by $\mu(x_i) = A_i$ for each $i \in [n]$ and $\mu(\epsilon) = I_s$, and vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{F}^s$ such that for every word $w \in \mathrm{X}^*$ we have $[w]S = \boldsymbol{u}^T \mu(w) \boldsymbol{v}$. For more details, see [3].

Now, we recall the definition of **multi-tape automata** (both deterministic and non-deterministic) following Worrell's work [20]. Let M be the pc monoid over variables $\mathrm{X} = \mathrm{X}_1 \cup \ldots \cup \mathrm{X}_k$ 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 $xy = yx$. A multi-tape automaton over M is a quintuple $A = (Q, \mathrm{X}, Q_0, F, E)$, where Q is a finite set of states, $E \subseteq Q \times \mathrm{X} \times Q$ is a set of edges, $Q_0 \subseteq Q$ are initial states, and $Q_f \subseteq Q$ are final states. A *run* of A from state q_0 to state q_m is a finite sequence of edges $e_1 e_2 \cdots e_m$ such that $e_i = (q_{i-1}, x_i, q_i)$ where each $x_i \in \mathrm{X}_j$ for some j. The label of the run is the product $x_1 x_2 \cdots x_m$ in the pc monoid M. Define the multiplicity $[m]A$ for some $m \in M$ to be the number of runs with label m such that $q_0 \in Q_0$ and $q_m \in Q_f$. More generally, in a $\mathbb{F}$-weighted k-tape automata, each transition is labelled with a scalar in $\mathbb{F}$. The weight of a run is the product of the weights of the edges in the run. The coefficient of any $m \in M$ is the sum of the weights of each accepting run for m.

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

Arithmetic Circuit Complexity : An *algebraic branching program* (ABP) is a directed acyclic graph with one in-degree-0 vertex called *source*, and one out-degree-0 vertex called *sink*. The vertex set of the graph is partitioned into layers $0, 1, \ldots, \ell$, with directed edges only between adjacent layers (i to $i+1$). The source and the sink are at layers zero and ℓ respectively. Each edge is labeled by an affine linear form over $\mathbb{F}$. 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 $\boldsymbol{u}^T M_1 M_2 \cdots M_\ell \boldsymbol{v}$, where $\boldsymbol{u}, \boldsymbol{v}$ 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 ℓ corresponds to the number of layers in the ABP. If X is a set of non-commuting variables then the ABP is a non-commutative algebraic branching program (e.g., see [14]).

Now we recall some results from non-commutative polynomial identity testing. Let $S \subset \mathbb{F}\langle \mathrm{X} \rangle$ be a subset of polynomials in the free non-commutative ring $\mathbb{F}\langle \mathrm{X} \rangle$ where $\mathrm{X} = \{x_1, \ldots, x_n\}$. Given a mapping $v : \mathrm{X} \to \mathcal{M}_t(\mathbb{F})$ 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 $v(f) = f(v(x_1), \ldots, v(x_n))$. A collection H of such evaluation maps is a *hitting set* for S, if for every nonzero f in S, there is an evaluation $v \in H$ such that $v(f) \neq 0$.

Let $S_{n,d,s}$ denote the subset of polynomials f in $\mathbb{F}\langle \mathrm{X} \rangle$ such that f has an algebraic branching program of size s and d layers. Forbes and Shpilka [8] have shown that a hitting set $H_{n,d,s}$ of quasi-polynomial size for $S_{n,d,s}$ can be constructed in quasi-polynomial time.

▶ **Theorem 5** (Forbes-Shpilka). *For all $s, d, n \in \mathbb{N}$ if $|\mathbb{F}| \geq \mathrm{poly}(d, n, r)$, then there is a set $H_{n,d,s}$ which is a hitting set for $S_{n,d,s}$. Further $|H_{n,d,s}| \leq (sdn)^{O(\log d)}$ and there is a deterministic algorithm to output the set $H_{n,d,s}$ in time $(sdn)^{O(\log d)}$.*

3 A Schützenberger Type Theorem for pc monoids

Let R be a subring of some division ring. A well-known theorem of Schützenberger [16] states that the series S computed by an R-weighted automaton of size s is nonzero if and only if for some word w of length at most $s-1$ its coefficient $[w]S$ in the series is nonzero. Worrell [20] uses this connection to reduce the multiplicity equivalence testing problem for k-tape automata to an instance of non-commutative polynomial identity testing.

In this section we obtain such a result for general partially commutative monoids.

First, we discuss partially commutative monoids and state the multiplicity equivalence problem over such monoids. Let X be a finite alphabet (equivalently, variable set) and $I \subseteq \mathrm{X} \times \mathrm{X}$ be a symmetric binary relation such that $(x_1, x_2) \in I$ if and only if $x_1 x_2 = x_2 x_1$. This relation I induces a *congruence* $\tilde{I}$ on X^* as follows: the words $m_1 x_1 x_2 m_2$ and $m_1 x_2 x_1 m_2$ are $\tilde{I}$-equivalent for $m_1, m_2 \in \mathrm{X}^*$ and $(x_1, x_2) \in I$. The transitive closure of this equivalence defines the congruence $\tilde{I}$. A *partially commutative monoid* (pc monoid for short) is defined as a pair $M = (X^*, I)$, where the monoid elements are defined as the congruence classes $\bar{m}$ for $m \in \mathrm{X}^*$ induced by $\tilde{I}$, and the monoid operation is concatenation. 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 $(x_1, x_2) \notin I$. It turns out that the structure of the graph G_M plays a significant role in multiplicity equivalence testing of automata over M.

A *k-partitioned pc monoid* is a partially commutative monoid for which the non-commutation graph can be partitioned into k vertex disjoint subgraphs.

Suppose a weighted automaton A is computing a series S over the pc monoid $M = (\mathrm{X}^*, I)$ and a series S' over the free monoid X^*. Notice that for each $\bar{m} \in M$ we have $[\bar{m}]S = \sum_{w\tilde{I}m}[w]S'$ where $w \in \mathrm{X}^*$. Automata A and B computing series S_A and S_B respectively over pc monoid M are *multiplicity equivalent* if $[\bar{m}]S_A = [\bar{m}]S_B$ for every $\bar{m} \in M$.

Embedding pc monoids into partitioned pc monoids :

We now show a transfer theorem that reduces the multiplicity equivalence problem over any general pc monoid to the multiplicity equivalence problem of an associated monoid which has a tensor product structure. This transformation is crucial for our algorithmic results.

For a field $\mathbb{F}$, let $\mathbb{F}\langle M\rangle$ be the $\mathbb{F}$-algebra of polynomials over a pc monoid $M = (\mathrm{X}^*, I)$. We will use interchangeably the terms *monomial* and *word* for elements of M. Let $\{G_i = (\mathrm{X}_i, E_i)\}_{i=1}^k$ be *any* covering of size k for the non-commutation graph $G_M = (\mathrm{X}, E)$: $\bigcup_i \mathrm{X}_i = \mathrm{X}$ and $\bigcup_i E_i = E$. Let $M_j = (\mathrm{X}_j^*, I_j)$ be the monoid with non-commutation graph G_j.

▶ **Lemma 6.** *Consider the map $\psi_1 : \mathbb{F}\langle M\rangle \to \mathbb{F}\langle M_1\rangle \otimes \cdots \otimes \mathbb{F}\langle M_k\rangle$ defined for each monomial $m \in M$ as : $\psi_1(m) = m_1 \otimes \cdots \otimes m_k$, where for each $i \in [k]$, $m_i = m|_{\mathrm{X}_i}$ and extended by linearity to $\mathbb{F}\langle M\rangle$. Then*

1. *ψ_1 is an injective homomorphism from the ring $\mathbb{F}\langle M\rangle$ to the ring $\mathbb{F}\langle M_1\rangle \otimes \cdots \otimes \mathbb{F}\langle M_k\rangle$.*
2. *If M is a k-partitioned pc monoid with respect to the partition $\{G_1, G_2, \ldots, G_k\}$ then the map ψ_1 is a ring isomorphism.*

Proof. The main property to check is that ψ_1 distributes over products. Let $m, m' \in M$. As $mm'|_{\mathrm{X}_i} = m|_{\mathrm{X}_i} \cdot m'|_{\mathrm{X}_i}$ [2], it follows that $\psi_1(mm') = \psi_1(m) \cdot \psi_1(m')$. It is now easy to check that ψ_1 is a ring homomorphism.

We now claim that $\psi_1(m) = \psi_1(m')$ implies $m = m'$ in M for any $m, m' \in M$. Indeed, this would imply for any nonzero $f \in \mathbb{F}\langle M\rangle$ that $\psi_1(f) \neq 0$. 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 $\psi_1(m) \neq \psi_1(m')$. The base case, for $\ell = 1$ clearly holds.

Now, suppose $m = x \cdot m_1 \in \mathrm{X}^{\ell+1}$ for $x \in \mathrm{X}$ and $\psi_1(m) = \psi_1(m')$.

▷ Claim 7. 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 $\psi_1(m) \neq \psi_1(m')$ which is a contradiction.

On other hand, suppose x occurs in m' and it cannot be moved to the leftmost position in m' using 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, $\psi_1(m) \neq \psi_1(m')$ which is a contradiction. ◀

Now, $\psi_1(x \cdot m_1) = \psi_1(x \cdot m_2)$ implies that $\psi_1(m_1) = \psi_1(m_2)$. Both m_1 and m_2 are of length ℓ. By induction hypothesis it follows that $m_1 = m_2$, and hence $m = m'$.

For the second part, we need to show that ψ_1 is surjective when the graphs G_j are pairwise disjoint. By linearity of ψ_1, it suffices to prove that each monomial $m_1 \otimes \cdots \otimes m_k$

[2] $m|_{\mathrm{X}_\mathrm{i}}$ is the restriction of the word m to the set X_i.

in $\mathbb{F}\langle M_1\rangle \otimes \cdots \otimes \mathbb{F}\langle M_k\rangle$ has a pre-image. As the graphs G_j are pairwise disjoint, the variable sets X_j are pairwise disjoint and there are no edges between X_j and X_ℓ for $j \neq \ell$. Hence, the pre-image of $m_1 \otimes \cdots \otimes m_k$ is the product $m_1 \cdot m_2 \cdots m_k$ which completes the proof. ◀

Associated Partitioned pc monoids :

Let $M = (\mathrm{X}^*, I)$ be a pc monoid. Suppose $\{G_i\}_{i=1}^k$ is a k-covering of the non-commutation graph G_M. That is, $G_i = (\mathrm{X}_i, E_i), 1 \le i \le k$, such that $\mathrm{X} = \bigcup_i \mathrm{X}_i$ and $E = \bigcup_i E_i$.

We define an associated monoid M' by defining its non-commutation graph $G_{M'} = (\mathrm{X}', E')$. For each $x_i \in \mathrm{X}$, let $J_i = \{j : x_i \in \mathrm{X}_j\}$. The vertex set X' of $G_{M'}$ is defined as $\{x_{ij} \mid x_i \in \mathrm{X}, j \in J_i\}$.

Thus, the graph $G_{M'}$ is essentially defined by splitting every vertex x_i of G_M into $|J_i|$ copies. The pair $(x_{ij}, x_{\ell j}) \in E'$ if and only if $j \in J_i \cap J_\ell$ and $(x_i, x_\ell) \in E$. This construction by splitting vertex x_i of G_M into $|J_i|$ copies makes $G_{M'}$ the union of k *disjoint* subgraphs $G'_j, 1 \le j \le k$. The monoid M' is defined by its non-commutation graph $G_{M'}$. For each j, M' has a submonoid M'_j on X'_j with non-commutation graph G'_j. The monoid M' is generated by the union of the submonoids $M'_1, \ldots, M'_k$.

For each $j \in [k]$, define the map $f_j : \mathrm{X} \to \mathrm{X}' \bigcup \{1\}$ as $f_j(x_i) = x_{ij}$ if $j \in J_i$, and $f_j(x_i) = 1$ otherwise. The maps f_j homomorphically extend to polynomials. Notice that f_j restricted to X_j is injective and $\mathrm{X}'_j = \{f_j(x_i) : x_i \in \mathrm{X}_j\}$.

Given a pc monoid M and a k-covering of its non-commutation graph G_M, we will show that multiplicity equivalence over M can be efficiently reduced to multiplicity equivalence over the associated partitioned pc monoid M'. To this end we first show a lemma. Following the notation of Lemma 6, recall that the monoids $M_1, \ldots, M_k$ correspond to the k-covering of M.

▶ **Lemma 8.** *The map $\psi_2 : \mathbb{F}\langle M_1\rangle \otimes \cdots \otimes \mathbb{F}\langle M_k\rangle \to \mathbb{F}\langle M'_1\rangle \otimes \cdots \otimes \mathbb{F}\langle M'_k\rangle$ defined for monomials $(m_1 \otimes \cdots \otimes m_k)$ as $\psi_2(m_1 \otimes \cdots \otimes m_k) = f_1(m_1) \otimes \cdots \otimes f_k(m_k)$ and extended by linearity to $\mathbb{F}\langle M_1\rangle \otimes \cdots \otimes \mathbb{F}\langle M_k\rangle$ is an injective ring homomorphism* [3].

The proof is simple and given in Section B of the appendix. An immediate corollary is the following.

▶ **Corollary 9.** *There is an injective homomorphism $\psi : \mathbb{F}\langle M\rangle \to \mathbb{F}\langle M'_1\rangle \otimes \cdots \otimes \mathbb{F}\langle M'_k\rangle$ defined as $\psi = \psi_2 \circ \psi_1$.*

Proof. As the composition of two injective homomorphisms is an injective homomorphism, the proof follows from Lemma 6 and Lemma 8. ◀

Now, combining Lemma 6 and Corollary 9, for any pc monoid M and a k-covering for G_M we get an explicit injective homomorphism from the ring $\mathbb{F}\langle M\rangle$ to the ring $\mathbb{F}\langle M'\rangle$, where M' is the associated partitioned pc monoid M'. Figure 1 shows the commutative diagram of the concerned maps. Here ψ'_1 is an isomorphism from part 2 of Lemma 6.

▶ **Lemma 10.** *Let M be any pc monoid with a k-covering of its non-commutation graph G_M and M' be the associated partitioned pc monoid. Then, there is an injective homomorphism $\varphi : \mathbb{F}\langle M\rangle \to \mathbb{F}\langle M'\rangle$ defined as $\varphi = \psi_1'^{-1} \circ \psi$. Moreover, $\varphi(x_i) = \prod_{j \in J_i} x_{ij}$.*

Proof. As ψ is an injective homomorphism (from Corollary 9) and $\psi_1'^{-1}$ is an isomorphism (from part 2 of Lemma 6), their composition φ is an injective homomorphism. Further note

[3] Indeed, the map ψ_2 is surjective, but we do not need the surjectivity in our proofs.

$$\begin{tikzcd}
\mathbb{F}\langle M\rangle \arrow[r, "\psi_1"] \arrow[d, "\varphi"'] & \mathbb{F}\langle M_1\rangle \otimes \cdots \otimes \mathbb{F}\langle M_k\rangle \arrow[d, "\psi_2"] \\
\mathbb{F}\langle M'\rangle & \mathbb{F}\langle M_1'\rangle \otimes \cdots \otimes \mathbb{F}\langle M_k'\rangle \arrow[l, "\psi_1'^{-1}"]
\end{tikzcd}$$

Figure 1 Commutative diagram illustrating chain of injective homomorphisms

that, $\psi(x_i) = f_1(x_i) \otimes \ldots \otimes f_k(x_i)$ which is the same as $(f_1(x_i) \otimes 1 \otimes \cdots \otimes 1) \cdot (1 \otimes f_2(x_i) \otimes \cdots \otimes 1) \cdots (1 \otimes 1 \otimes \cdots \otimes f_k(x_i))$. Therefore, $\varphi(x_i) = \psi_1'^{-1}(\psi(x_i)) = f_1(x_i)f_2(x_i)\cdots f_k(x_i)$. Hence, $\varphi(x_i) = \prod_{j \in J_i} x_{ij}$. ◀

Reduction to Equivalence Testing over Partitioned Monoid :

We next show that multiplicity equivalence of automata over any pc monoid M whose non-commutation graph G_M has a k-covering can be reduced to multiplicity equivalence testing of automata over the associated partitioned pc monoid M'.

▶ **Lemma 11.** *Let A and B be $\mathbb{F}$-weighted automata of size s over a partially commutative monoid $M = (\mathrm{X}^*, I)$, for which the non-commutation graph G_M has an m-covering $\{G_i = (\mathrm{X}_i, E_i)\}_{i=1}^m$. Then, multiplicity equivalence of A and B is reducible in* poly(s) *time to multiplicity equivalence of $\mathbb{F}$-weighted automata A' and B' of total size at most ns^2m over the associated partitioned pc monoid M'.*

The proof of the lemma is in Section B of the appendix. The essential idea is to obtain the automata A' and B' over M' by replacing each variable x_i by its image $\varphi(x_i)$. It is also crucially used that φ is an injective homomorphism which we have proved in Lemma 10.

Now we slightly restate a result of Worrell [20] that shows that the multiplicity equivalence problem over partitioned pc monoid for which the non-commutation graph is a disjoint union of cliques, can be reduced to non-commutative polynomial identity testing. Worrell uses a truncation argument following the result of Schützenberger [16].

▶ **Proposition 12** (Restatement of Proposition 5 of [20]). *Let M' be a partitioned pc monoid whose non-commutation graph $G_{M'}$ is a disjoint union of m cliques. Then $\mathbb{F}$-weighted automata $A' = (Q_{A'}, \mathrm{X}', E_{A'}, \boldsymbol{u_{A'}}, \boldsymbol{v_{A'}}, wt_{A'})$ of size s_1 and $B' = (Q_{B'}, \mathrm{X}', E_{B'}, \boldsymbol{u_{B'}}, \boldsymbol{v_{B'}}, wt_{B'})$ of size s_2 over M' are multiplicity equivalent, if and only if, $\boldsymbol{u}'^T \hat{N}^\ell \boldsymbol{v}' = 0$ for each $0 \leq \ell \leq s-1$, where $s = s_1 + s_2$. The matrices are,*

$$\boldsymbol{u'} = \begin{bmatrix} \boldsymbol{u}'_A \\ \boldsymbol{u}'_B \end{bmatrix}, \qquad \hat{N} = \begin{bmatrix} N_{A'} & 0 \\ 0 & N_{B'} \end{bmatrix}, \qquad \boldsymbol{v'} = \begin{bmatrix} \boldsymbol{v}'_A \\ -\boldsymbol{v}'_B \end{bmatrix}.$$

The matrices $N_{A'}, N_{B'}$ are the weighted unary matrices for the automata A' and B'. More precisely, $1 \leq i,j \leq s_1 : N_{A'}[i,j] = \sum_{e \in E_{A'}:e=(q_i,x,q_j)} wt'(e) \cdot x$. ($N_{B'}$ is defined similarly).

Notice that, from the definition of algebraic branching programs in Section 2, $\boldsymbol{u}'^T \hat{N}^\ell \boldsymbol{v}'$ for each $0 \leq \ell \leq s-1$, can be represented by small-size algebraic branching programs of width s and number of layers bounded by ℓ. Lemma 11 and Proposition 12 yield the following generalization of Schützenberger's theorem [16] over arbitrary partially commutative monoids.

▶ **Theorem 13.** *Let A and B be two $\mathbb{F}$-weighted automata of total size s over any pc monoid $M = (\mathrm{X}^*, I)$. Deciding multiplicity equivalence of A and B can be reduced in* poly(s) *time to identity testing of a collection of algebraic branching programs $\{B_d\}_{d=0}^D$ computing polynomials* *in* $\mathbb{F}\langle M\rangle$ *where* $D \le s^2n^3$ *and the width of each* B_d *is at most* s *having* d *many layers. For each* d*,* B_d *is defined as* $\boldsymbol{u}^T N^d \boldsymbol{v}$*, where the vectors* $\boldsymbol{u}, \boldsymbol{v}$ *and the transition matrix* N *are obtained from the input automata* A *and* B *in a similar way as described in Proposition 12.*

In particular, if the input monoid M is given by a clique cover of size m, we get a bound of $D \le ns^2m$. Now, we sketch the proof briefly (a detailed proof is in Section B of the appendix). First we choose a clique cover of size $m \le n^2$ for the non-commutation graph G_M. We apply Lemma 11 with respect to this clique cover and construct automata A' and B' from A and B over the associated partitioned pc monoid M' (whose non-commutation graph is a disjoint union of cliques). By Proposition 12, multiplicity equivalence of A' and B' is reducible to testing the identities of the algebraic branching programs $\boldsymbol{u'}^T \hat{N}^d \boldsymbol{v'} = 0$ for each $0 \le d \le ns^2m \le s^2n^3$. As the map φ (of Lemma 10) is injective, it suffices to check the identities of the algebraic branching programs $\boldsymbol{u}^T N^d \boldsymbol{v} = 0$ for each $0 \le d \le s^2n^3$ constructed from the weighted unary matrices corresponding to A and B. The matrix N is obtained from the unary matrices N_A and N_B in the same way as $\hat{N}$ is constructed from N'_A and N'_B.

4 Deterministic Multiplicity Equivalence Test over k-Clique Monoids

A *k-clique monoid* is a pc monoid M whose non-commutation graph G_M has a clique cover of size k. In this section, we show that the multiplicity equivalence problem over k-clique monoids for constant k can be solved in deterministic quasi-polynomial time.

For each $i \in [k]$, let us define $\mathrm{X}'_i = \{x_{1i}, \ldots, x_{n_i i}\}$. Now, $\mathrm{X}'_1, \ldots, \mathrm{X}'_k$ are disjoint sets of total $\sum_{i=1}^k n_i = \tilde{n}$ many variables. Suppose $\mathrm{X}' = \bigcup_{i=1}^k \mathrm{X}'_i$. We first give an identity testing algorithm for polynomials in $\mathbb{F}\langle \mathrm{X}'_1\rangle \otimes \cdots \otimes \mathbb{F}\langle \mathrm{X}'_k\rangle$ which will be used to prove Theorem 1. Let us first define the notion of evaluation and partial evaluation of polynomials in $\mathbb{F}\langle \mathrm{X}'_1\rangle \otimes \cdots \otimes \mathbb{F}\langle \mathrm{X}'_k\rangle$.

Evaluation of a polynomial over algebras : Given 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 $\mathbf{A} = (A_1, \ldots, A_k)$, an *evaluation* of f in $\mathbf{A}$ 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 $\boldsymbol{v}(x) = I_1 \otimes \cdots \otimes v_j(x) \otimes \cdots \otimes I_k$ where I_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 $\mathbf{A}$. Let $k' < k$ and $\hat{\mathbf{A}} = (A_1, \ldots, A_{k'})$ 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 $\hat{\mathbf{A}}$ 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 $A_1 \otimes \cdots \otimes A_{k'} \otimes \mathbb{F}\langle \mathrm{X}'_{k'+1}\rangle \otimes \cdots \mathbb{F}\langle \mathrm{X}'_k\rangle$.

Now we generalize the definition of ABP over $\mathbb{F}\langle \mathrm{X}'_1\rangle \otimes \cdots \otimes \mathbb{F}\langle \mathrm{X}'_k\rangle$. We note that each tensor of the form $1 \otimes \cdots \otimes x_{ij} \otimes \cdots \otimes 1$ plays the role of a variable in $\mathbb{F}\langle \mathrm{X}'_1\rangle \otimes \cdots \otimes \mathbb{F}\langle \mathrm{X}'_k\rangle$ and consequently the linear forms are linear combinations of such tensors. An algebraic branching program over $\mathbb{F}\langle \mathrm{X}'_1\rangle \otimes \cdots \otimes \mathbb{F}\langle \mathrm{X}'_k\rangle$ has edges labelled by such affine linear forms.

Let $S_{k,\tilde{n},d,s}$ 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 algebraic branching programs of size s and d layers, where $\tilde{n}$ is the total number of variables. Following the notation in Theorem 5, let $\mathcal{H}_{k,\tilde{n},d,s}$ be a hitting set for $S_{k,\tilde{n},d,s}$. That is, $\mathcal{H}_{k,\tilde{n},d,s}$ is a collection of evaluations $\boldsymbol{v} = (v_1, \ldots, v_k)$, where each $v_i : \mathrm{X}_i \to \mathcal{M}_{d+1}(\mathbb{F})$, such that for any nonzero polynomial $f \in S_{k,\tilde{n},d,s}$ there is an evaluation $\boldsymbol{v} = (v_1, \ldots, v_k) \in \mathcal{H}_{k,\tilde{n},d,s}$ such that $\boldsymbol{v}(f)$ is a nonzero matrix of dimension $(d+1)^k$ over $\mathbb{F}$. For $k = 1$, Forbes and Shpilka [8] have constructed a quasi-polynomial size hitting set $\mathcal{H}_{1,\tilde{n},d,s}$. (see Theorem 5). The following key lemma shows an efficient bootstrapped construction of a hitting set $\mathcal{H}_{k,\tilde{n},d,s}$ for the set $S_{k,\tilde{n},d,s}$ of polynomials, using the hitting set $\mathcal{H}_{1,\tilde{n},d,s}$.

▶ **Lemma 14.** *There is a set of evaluation maps $\mathcal{H}_{k,\tilde{n},d,s} = \{(v_1, \ldots, v_k) : v_i \in \mathcal{H}_{1,\tilde{n},d,s_k}\}$ where $s_k = s(d+1)^{2(k-1)}$ such that, for $i \in [k]$, we have $v_i : \mathrm{X}'_i \to \mathcal{M}_{d+1}(\mathbb{F})$, and $\mathcal{H}_{k,\tilde{n},d,s}$ is a hitting set for the class of polynomials $S_{k,\tilde{n},d,s}$. Moreover, the size of the set is at most $(\tilde{n}skd)^{O(k^2 \log d)}$, and it can be constructed in deterministic $(\tilde{n}skd)^{O(k^2 \log d)}$ time.*

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 $f = \sum_{m \in \mathrm{X}'^*_k} f_m \otimes m$ 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 15.** *For each $f \in S_{k,\tilde{n},d,s}$ 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 algebraic branching program of size $s \cdot (d+1)^2$ and d many 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 $m = x_{i_1 k} x_{i_2 k} \ldots x_{i_\ell k}$ where some of the indices could be repeated. Let $M' = (\mathrm{X}'^*, I)$ be a partitioned pc monoid where for each i, j, variables in X'_i and X'_j commute but for each i, variables in X'_i do not commute. In other words, the non-commutation graph $G_{M'}$ is the disjoint union of cliques over $\mathrm{X}'_1, \ldots, \mathrm{X}'_k$. Recall that, $\psi'_1 : \mathbb{F}\langle M' \rangle \to \mathbb{F}\langle \mathrm{X}'_1 \rangle \otimes \cdots \otimes \mathbb{F}\langle \mathrm{X}'_k \rangle$ is an isomorphism as defined in part 2 of Lemma 6. Let us define an ABP ${\psi'_1}^{-1}(B)$ by applying ${\psi'_1}^{-1}$ on each edge of B labelled by a linear form over variables of form $1 \otimes \cdots \otimes x_{ij} \otimes \cdots \otimes 1$. From the definition, ${\psi'_1}^{-1}$ substitutes each $1 \otimes \cdots \otimes x_{ij} \otimes \cdots \otimes 1$ with x_{ij}. Notice that ${\psi'_1}^{-1}(B)$ computes ${\psi'_1}^{-1}(f)$ in $\mathbb{F}\langle M' \rangle$. Let ${\psi'_1}^{-1}(B)$ over the *free* non-commutative ring $\mathbb{F}\langle \mathrm{X}' \rangle$ computing a polynomial g. Recall that, for each $m' \in M'$, $[m']f = \sum_{w \tilde{I} m'} [w]g$. Now, we construct an automaton A that isolates precisely those words (monomials) $w \in \mathrm{X}'^*$ from g such that $w|_{\mathrm{X}'_k} = m$. The automaton A is depicted in Figure 2.

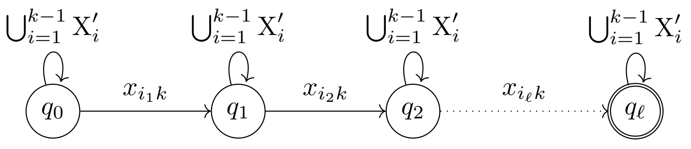
$\bigcup_{i=1}^{k-1} \mathrm{X}'_i$
$\bigcup_{i=1}^{k-1} \mathrm{X}'_i$
$\bigcup_{i=1}^{k-1} \mathrm{X}'_i$
$\bigcup_{i=1}^{k-1} \mathrm{X}'_i$
q_0
$x_{i_1 k}$
q_1
$x_{i_2 k}$
q_2
$x_{i_\ell k}$
q_ℓ

Figure 2 The transition diagram of the automaton A

The automaton simply loops around in each state q_t if the input letter is in $\bigcup_{i=1}^{k-1} \mathrm{X}'_i$. It makes a forward transition from q_t to q_{t+1} only on reading $x_{i_{t+1} k}$, for $0 \le t \le \ell - 1$.

Naturally, the ABP ${\psi'_1}^{-1}(B)$ can be thought of as a $\mathbb{F}$-weighted acyclic automaton B' without any ϵ-transition [4] computing same g. Now we compute the Hadamard product of B' with A, denoted by $B' \odot A$ over the *free* monoid computing $g \odot A$. By a basic result of Schützenberger [15, Theorem 3.2, pp. 428], it is known that $B' \odot A$ has an automaton of size $s \cdot (\ell + 1)$. Let M be the unary transition matrix of this automaton whose entries are linear forms over X'. Let $\boldsymbol{u}$, $\boldsymbol{v}$ be the weight vectors for initial and final states. The monomials of degree d can be computed by an ABP $\hat{B}$ of size $s(\ell + 1)d$ and d many layers

[4] In fact any ABP can be also represented by an weighted acyclic automaton of similar size, such that the polynomial computed by the ABP and the finite series computed by the automaton are the same.

defined as $\hat{B} = \boldsymbol{u}^T M^d \boldsymbol{v}$. We can simply substitute X'_k variables by 1. Now, we take the ψ'_1 image of $\hat{B}|_{X'_k=1}$ (defined similarly by applying ψ'_1 on each edge of $\hat{B}$) to get the required ABP $\psi'_1(\hat{B}|_{X'_k=1})$ of the same size computing the polynomial f_m. Since $\ell \leq d$, the lemma follows. ◀

For a polynomial f in $\mathbb{F}\langle X'_1\rangle \otimes \cdots \otimes \mathbb{F}\langle X'_k\rangle$, consider a partial evaluation $\boldsymbol{v} = (v_1, \ldots, v_{k-1})$ such that each $v_i : X'_i \to \mathcal{M}_{t_i}(\mathbb{F})$. The evaluation $\boldsymbol{v}(f)$ is a $T \times T$ matrix with entries from $\mathbb{F}\langle X'_k\rangle$, and $T = t_1 t_2 \cdots t_{k-1}$.

▶ **Lemma 16.** *For each $p, q \in [T]$, the $(p,q)^{th}$ entry of $\boldsymbol{v}(f)$ can be computed by an algebraic branching program of size sT and d layers.*

The proof is given in Section C of the appendix. Now we are ready to prove Lemma 14.

Proof of Lemma 14. The proof is by induction on k. For the base case $k = 1$ the hitting set $\mathcal{H}_{1,\tilde{n},d,s}$ from Theorem 5 suffices. Note that any nonzero $f \in S_{k,\tilde{n},d,s}$ can be written as $f = \sum_{m \in X'^*_k} f_m \otimes m$ where each m is a monomial over X'_k and $f_m \in \mathbb{F}\langle X'_1\rangle \otimes \cdots \otimes \mathbb{F}\langle X'_{k-1}\rangle$. Since $f \not\equiv 0$ we must have $f_m \not\equiv 0$ for some $m \in X'^*_k$. Moreover, by Lemma 15 we know that for each $m \in {X'_k}^*$ the polynomial $f_m \in \mathbb{F}\langle X'_1\rangle \otimes \cdots \otimes \mathbb{F}\langle X'_{k-1}\rangle$ can be computed by an algebraic branching program of size $s \cdot (d+1)^2$.

Now, inductively assume we have the hitting set

$$\mathcal{H}_{k-1,\tilde{n},d,s'} = \{(v_1, v_2, \ldots, v_{k-1}) | v_i \in \mathcal{H}_{1,\tilde{n},d,s'_{k-1}}\}$$

for the class of polynomials $S_{k-1,\tilde{n},d,s'}$ where $s' = s(d+1)^2$. By the inductive hypothesis, $s'_{k-1} = s'(d+1)^{2(k-2)} = s(d+1)^{2(k-1)}$ and $v_i : X'_i \mapsto \mathcal{M}_{d+1}(\mathbb{F})$. Since $f_m \not\equiv 0$ and $f_m \in S_{k-1,\tilde{n},d,s(d+1)^2}$, there is an evaluation $\boldsymbol{v'} \in \mathcal{H}_{k-1,\tilde{n},d,s(d+1)^2}$ such that $\boldsymbol{v'}(f_m)$ is a non-zero matrix of dimension $(d+1)^{k-1} \times (d+1)^{k-1}$. 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 $\mathbb{F}\langle X'_k\rangle$. Since $\boldsymbol{v'}(f_m) \neq 0$, it follows that some $(p,q)^{th}$ entry of $\boldsymbol{v'}(f)$ is a nonzero polynomial $g \in \mathbb{F}\langle X'_k\rangle$. By Lemma 16, each entry of $\boldsymbol{v'}(f)$ has an algebraic branching program of size[5] $s(d+1)^{2(k-1)}$. In particular, $g \in S_{1,\tilde{n},d,s(d+1)^{2(k-1)}}$ and it follows from Theorem 5 that there is a an evaluation v'' in $\mathcal{H}_{1,\tilde{n},d,s(d+1)^{2(k-1)}}$ such that $v''(g)$ is a non-zero matrix of dimension $(d+1) \times (d+1)$. Thus, for the combined evaluation map $\boldsymbol{v} = (\boldsymbol{v'}, v'')$, it follows that $\boldsymbol{v}(f)$ is a non-zero matrix of dimension $(d+1)^k \times (d+1)^k$. Define $\mathcal{H}_{k,\tilde{n},d,s} = \{(v_1, \ldots, v_k) : v_i \in \mathcal{H}_{1,\tilde{n},d,s_k}\}$, where $s_k = s \cdot (d+1)^{2(k-1)}$. However, from the inductive hypothesis, we know that $\boldsymbol{v'} = (v_1, \ldots, v_{k-1}) \in \mathcal{H}_{k-1,\tilde{n},d,s(d+1)^2}$ where each $v_i \in \mathcal{H}_{1,\tilde{n},d,s(d+1)^{2(k-1)}}$. Therefore, $\boldsymbol{v} = (\boldsymbol{v'}, v'') \in \mathcal{H}_{k,\tilde{n},d,s}$ and $\mathcal{H}_{k,\tilde{n},d,s}$ is a hitting set for the class of polynomials $S_{k,\tilde{n},d,s}$.

Finally, note that $|\mathcal{H}_{k,\tilde{n},d,s}| = |\mathcal{H}_{1,\tilde{n},d,s_k}|^k$. Since $|\mathcal{H}_{1,\tilde{n},d,s_k}| \leq (\tilde{n} d s_k)^{O(\log d)}$, it follows that $|\mathcal{H}_{k,\tilde{n},d,s}| \leq (\tilde{n} s k d)^{O(k^2 \log d)}$. Clearly, the set $\mathcal{H}_{k,\tilde{n},d,s}$ can be constructed in the claimed running time.

4.1 Proof of Theorem 1

Proof. Given two $\mathbb{F}$-weighted automata A and B of total size s over a k-clique monoid $M = (X^*, I)$ as input, by Theorem 13, multiplicity equivalence of A and B reduces to identity testing of a collection of algebraic branching programs $\{B_d\}_{d=1}^{n^3 s^2}$ over $\mathbb{F}\langle M\rangle$, where each B_d is of size sd and has d layers. Let us fix a d and suppose B_d is computing a polynomial f in $\mathbb{F}\langle M\rangle$.

[5] For the ease of calculation we use a weaker bound even though Lemma 16 gives a bound of $s(d+1)^{k-1}$.

Now, recall from Section 3 that given a k-clique monoid M, we can construct M', the associated partitioned pc monoid, with the disjoint components $\{M'_i\}_{i=1}^k$ defined over disjoint variable sets $\{\mathrm{X}'_i\}_{i=1}^k$, where each $G_{M'_i}$ is a clique. Hence the $\mathbb{F}$-algebra $\mathbb{F}\langle M'_j\rangle$ can be identified with $\mathbb{F}\langle \mathrm{X}'_j\rangle$. Now the idea is to apply the map ψ on B_d to get algebraic branching program $\psi(B_d)$ over $\mathbb{F}\langle M'_1\rangle \otimes \cdots \otimes \mathbb{F}\langle M'_k\rangle$. Since ψ is injective from Corollary 9, it suffices to test the identity of $\psi(B_d)$. To obtain the branching program $\psi(B_d)$, we simply replace the variables x_i on each edge of B by a k-length path labelled as $(f_1(x_i) \otimes 1 \otimes \cdots \otimes 1) \cdot (1 \otimes f_2(x_i) \otimes \cdots \otimes 1) \cdots (1 \otimes 1 \otimes \cdots \otimes f_k(x_i))$. A routine calculation shows that $\psi(B_d)$ has an ABP of size at most $t_d = (n+1)ks^2d$, with at most kd layers. The details are given in Section C of the appendix.

Now, for each d, we construct the hitting set $\mathcal{H}_{k,nk,kd,t_d}$ from Lemma 14. If $\boldsymbol{v}(B'_d)$ is the zero matrix for each evaluation $\boldsymbol{v} \in \mathcal{H}_{k,nk,kd,t_d}$ then we can conclude that $B'_d \equiv 0$ as $\mathcal{H}_{k,nk,kd,t_d}$ is a hitting set for B_d. A pseudocode description of the algorithm is in Section C of the appendix. ◀

5 Randomized Algorithm for Multiplicity Equivalence Testing of k-Monoid

We now consider pc monoids more general than k-clique monoids, over which too we can do efficient equivalence testing of automata. A *k-monoid* is a pc monoid M whose non-commutation graph G_M is a union of subgraphs $G_M = G_1 \cup G_2$ such that G_1 has a clique cover of size k' and G_2 has a vertex cover of size $k - k'$. It follows that G_M has a k-covering of cliques and star graphs. For the application to equivalence testing, we will assume that this k-covering of G_M is explicitly given as part of the input.

▶ **Lemma 17.** *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 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 $\boldsymbol{v} = (v_1, \ldots, v_k)$ 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 $1/2$.

The proof of this lemma is in Section D of the appendix.

For the proof of Theorem 2, we first give a randomized polynomial-time identity testing algorithm for polynomials over pc monoids whose non-commutation graph is a star graph.

▶ **Lemma 18.** *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 $v : \mathrm{X} \cup \{y\} \to \mathcal{M}_{t(d)}(\mathbb{F})$ 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 $v(f)$ 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 $\ell \leq d$. 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 3.

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

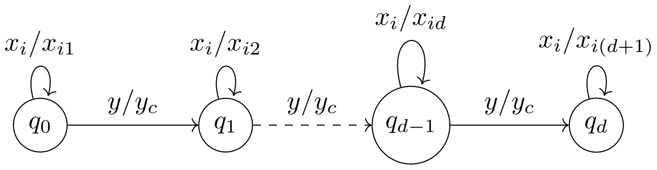

Figure 3 The transition diagram of the automaton

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. In effect, we substitute each x_i and y by the transition matrices N_{x_i} and N_y of dimension $d+1$. The transition matrices are explicitly given in Section D of the appendix.

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.

▶ **Observation 1.** *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 [21, 17, 6]. ◀

Now we are ready to prove Theorem 2.

Proof. 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 [1], 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.[6] 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 18.

Given two $\mathbb{F}$-weighted automata A and B of total size s over a k-monoid $M = (\mathrm{X}^*, I)$, by Theorem 13, multiplicity equivalence of A and B reduces to identity testing of a collection of algebraic branching programs $\{B_d\}_{d=1}^{n^3 s^2}$ over $\mathbb{F}\langle M\rangle$, where each B_d is of size sd and has d layers. Let us fix a d and suppose B_d is computing a polynomial $f \in \mathbb{F}\langle M\rangle$. 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 Corollary 9. 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 17, 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 $d \le s^2 n^3$ by Lemma 13. This completes the proof of Theorem 2. ◀

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

References

1 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.

2 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. URL: http://www.sciencedirect.com/science/article/pii/0304397576900499, doi:https://doi.org/10.1016/0304-3975(76)90049-9.

3 J. Berstel and C. Reutenauer. *Noncommutative Rational Series with Applications*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2011. URL: https://books.google.co.in/books?id=LL8Nhn72I_8C.

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

5 Andrej Bogdanov and Hoeteck Wee. More on noncommutative polynomial identity testing. In *20th Annual IEEE Conference on Computational Complexity (CCC 2005), 11-15 June 2005, San Jose, CA, USA*, pages 92–99, 2005. URL: https://doi.org/10.1109/CCC.2005.13, doi:10.1109/CCC.2005.13.

6 Richard A. Demillo and Richard J. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7(4):193 – 195, 1978. URL: http://www.sciencedirect.com/science/article/pii/0020019078900674, doi:https://doi.org/10.1016/0020-0190(78)90067-4.

7 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. URL: https://doi.org/10.1007/3-540-63165-8_222, doi:10.1007/3-540-63165-8_222.

8 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. URL: http://dx.doi.org/10.1109/FOCS.2013.34, doi:10.1109/FOCS.2013.34.

9 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.

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

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

12 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. URL: https://doi.org/10.1007/3-540-17906-2_30, doi:10.1007/3-540-17906-2_30.

13 Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7:105–113, 1987.

14 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. URL: http://doi.acm.org/10.1145/103418.103462, doi:10.1145/103418.103462.

15 Jacques Sakarovitch. *Elements of Automata Theory.* Cambridge University Press, 2009. doi:10.1017/CBO9781139195218.

16 M.P. Schützenberger. On the definition of a family of automata. *Information and Control*, 4(2):245 – 270, 1961. URL: http://www.sciencedirect.com/science/article/pii/S001999586180020X, doi:https://doi.org/10.1016/S0019-9958(61)80020-X.

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

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

19 Leslie G. Valiant. The equivalence problem for deterministic finite-turn pushdown automata. *Information and Control*, 25(2):123 – 133, 1974. URL: http://www.sciencedirect.com/science/article/pii/S0019995874908390, doi:https://doi.org/10.1016/S0019-9958(74)90839-0.

20 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. URL: https://doi.org/10.1007/978-3-642-39212-2_38, doi:10.1007/978-3-642-39212-2_38.

21 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 Proposition 4

Proof. Let A_i be the adjacency matrix of the automaton A keeping only the edges labelled by x_i for each i. Let A_0 be the adjacency matrix of the automaton A keeping only the edges labelled by ϵ-transitions. Let $\boldsymbol{u}, \boldsymbol{v}$ be the weighted vectors corresponding to the initial and final states. Then the series computed by the automaton $\mathcal{A}$ is

$$S = \boldsymbol{u}^T \cdot \left(\sum_{k=0}^{\infty} (A_0 + \sum_{i=1}^{n} A_i x_i)^k \right) \cdot \boldsymbol{v}$$

Since S is a formal series, we conclude that $\sum_{k=0}^{\infty} A_0^k$ must converge to the matrix $(I - A_0)^{-1}$. Now we define new transition matrices for the automaton B. For the variable x_i we construct the matrix $B_i = (I - A_0)^{-1} \cdot A_i$ and we modify $\boldsymbol{u'} = \boldsymbol{u}$ and $\boldsymbol{v'} = (I - A_0)^{-1} \cdot \boldsymbol{v}$.

For a word $w = x_{i_1} x_{i_2} \cdots x_{i_d}$ its coefficient in A is

$$\begin{aligned}
&\boldsymbol{u}^T \cdot \left(\sum_{j_1=0}^{\infty} \sum_{j_2=0}^{\infty} \cdots \sum_{j_{d+1}=0}^{\infty} A_0^{j_1} \cdot A_{i_1} \cdot A_0^{j_2} \cdot A_{i_2} \cdots A_0^{j_d} \cdot A_{i_d} \cdot A_0^{j_{d+1}} \right) \cdot \boldsymbol{v} \\
&= \boldsymbol{u}^T \cdot \left((\sum_{j_1=0}^{\infty} A_0^{j_1}) \cdot A_{i_1} \cdot (\sum_{j_2=0}^{\infty} A_0^{j_2}) \cdot A_{i_2} \cdots (\sum_{j_d=0}^{\infty} A_0^{j_d}) \cdot A_{i_d} \cdot (\sum_{j_{d+1}=0}^{\infty} A_0^{j_{d+1}}) \right) \cdot \boldsymbol{v} \\
&= \boldsymbol{u}^T \cdot \left((I - A_0)^{-1} \cdot A_{i_1} \cdot (I - A_0)^{-1} \cdot A_{i_2} \cdots (I - A_0)^{-1} \cdot A_{i_d} \cdot (I - A_0)^{-1} \right) \cdot \boldsymbol{v} \\
&= \boldsymbol{u}'^T \cdot (B_{i_1} \cdot B_{i_2} \cdots B_{i_d}) \cdot \boldsymbol{v}',
\end{aligned}$$

which is the same in the automaton B. By construction B is free of ϵ-transitions. ◀

B Missing Proofs from Section 3

B.1 The Proof of Lemma 8

Proof. To show that ψ_2 is a ring homomorphism we need to prove that ψ_2 preserves the ring product. Given monomials $m_1, m_2 \in M_1 \otimes \cdots \otimes M_k$, we show that $\psi_2(m_1 m_2) =$

$\psi_2(m_1)\cdot\psi_2(m_2)$. Write $m_1 = (m_{11}\otimes\cdots\otimes m_{k1})$ and $m_2 = (m_{12}\otimes\cdots\otimes m_{k2})$. Then, $\psi_2(m_1 m_2) = \psi_2(m_{11}m_{12} \otimes \cdots \otimes m_{k1}m_{k2}) = f_1(m_{11}m_{12}) \otimes \cdots \otimes f_k(m_{k1}m_{k2}) = \psi_2(m_1)\psi_2(m_2)$. By linearity, it follows that ψ_2 is a ring homomorphism.

To show that ψ_2 is injective, we need to prove that for each $m, m' \in M_1 \otimes \cdots \otimes M_k$, if $\psi_2(m) = \psi_2(m')$ then $m = m'$. Let $m = (m_{11} \otimes \cdots \otimes m_{k1})$ and $m' = (m_{12} \otimes \cdots \otimes m_{k2})$. As $\psi_2(m) = \psi_2(m')$, it must be the case that for each $j \in [k]$, $f_j(m_{j1}) = f_j(m_{j2})$. But for each $j \in [k]$, f_j is injective when restricted to X_j. Therefore, $m_{j1} = m_{j2}$ and $m = m'$. ◀

B.2 The Proof of Lemma 11

Proof. As already defined, let J_i be the set $J_i = \{j : x_i \in \mathrm{X}_j\}$. Let us replace each edge labelled as x_i by the product $\prod_{j\in J_i} x_{ij}$. More Precisely, given an $\mathbb{F}$-weighted automaton A of size s_1 over $M = (\mathrm{X}^*, I)$, we derive an $\mathbb{F}$-weighted automaton A' over M' in the following way.

For each $q_0, q_m \in Q$ such that $(q_0, x_i, q_m) \in E$ and $wt(q_0, x_i, q_m) = \alpha \in \mathbb{F}$, we introduce new states $q_1, \ldots, q_{m-1}$ and for each $j \le m - 1$, add

$$e_j = (q_{j-1}, f_j(x_i), q_j)$$

in E' and $wt'(e_1) = \alpha$ and for other edges $wt'(e_j) = 1$. The vectors $\boldsymbol{u}', \boldsymbol{v}'$ are obtained from $\boldsymbol{u}, \boldsymbol{v}$ by retaining the weights corresponding to the original states and zero for the newly added states. The size of A' is bounded by $ns_1^2 m$. A roundabout way to see this is the following. Between each two states in the original automata, there are at most n weighted edges, one for each x_i. Hence total number of edges can be bounded by ns_1^2. Then each such edge is replaced by a path of length at most m by introducing new states. Here Similarly, we define B'. Hence, the total size of A' and B' is bounded by ns^2m.

We claim that, A and B are multiplicity equivalent if and only if A' and B' are multiplicity equivalent. Let S_A and $S_{A'}$ be the formal series computed by the automata A and A' respectively. From Lemma 10, recall that $\varphi(x_i) = \prod_{j\in J_i} x_{ij}$. From the construction, the series $S_{A'}$ is obtained by replacing each x_i by $\varphi(x_i)$, i.e. $S_{A'} = S(\varphi(x_1), \ldots, \varphi(x_n)) = \varphi(S_A)$. Similarly, we define the series S_B and $S_{B'}$ and $S_{B'} = \varphi(S_B)$. For a series S, define $S^{(\ell)}$ to be the restriction of the series S to the words of length ℓ.

Now, A and B are multiplicity equivalent, if and only if, for each $\ell \in \mathbb{N}$, $S_A^\ell = S_B^\ell$. Following Lemma 10, since φ is injective, it is equivalent to checking for each $\ell \in \mathbb{N}$, $\varphi(S_A^\ell) = \varphi(S_B^\ell)$. But, for each $\ell \in \mathbb{N}$, $\varphi(S_A^\ell) = \varphi(S_B^\ell)$, if and only if, $S_{A'}$ and $S_{B'}$ are multiplicity equivalent, which is equivalent to A' and B' are multiplicity equivalent. ◀

B.3 The Proof of Theorem 13

Proof. Let the non-commutation graph G_M of $M = (\mathrm{X}^*, I)$ has a clique cover of size m, $\{\mathrm{X}_j\}_{j=1}^m$. Define M' to be the partitioned partially commutative monoid corresponding to M and the m-sized clique cover. Using Lemma 10, we construct two $\mathbb{F}$-weighted automata A' and B' of total size s^2mn over M' and it suffices to check the multiplicity equivalence of A' and B'. Now, by Proposition 12 the multiplicity equivalence testing of A' and B' reduces to the identity testing of algebraic branching programs $B'_d = \boldsymbol{u}'^T \hat{N}^d \boldsymbol{v}'$ for each $0 \le d \le s^2mn$. From Lemma 10, we know that φ is an injective homomorphism between $\mathbb{F}\langle M\rangle$ and $\mathbb{F}\langle M'\rangle$, thus for any $m \in M$ and any series S over $\mathbb{F}\langle M\rangle$ we have

$$[m]S = [\varphi(m)]\varphi(S).$$

Now if B'_d is not an identically zero polynomial, then we have $[m']B'_d \neq 0$ for some $m' \in M'$. From the construction, there is some $m \in M$, such that $m' = \varphi(m)$. Hence, $[\varphi(m)]B'_d \neq 0$. Further as m' has length d, m has length $\ell \leq d$ then $[m]B_\ell = [\varphi(m)]B'_d \neq 0$ (further $[m]B_{\ell'} = 0$ for $\ell \neq \ell'$) which shows that $B_\ell \not\equiv 0$. Thus it suffices to test that the branching programs $B_d \equiv 0$ for each $d \leq s^2 m$ and $B_d = \boldsymbol{u}^T N^d \boldsymbol{v}$ where

$$\boldsymbol{u} = \begin{bmatrix} \boldsymbol{u}_A \\ \boldsymbol{u}_B \end{bmatrix}, \qquad N = \begin{bmatrix} N_A & 0 \\ 0 & N_B \end{bmatrix}, \qquad \boldsymbol{v} = \begin{bmatrix} \boldsymbol{v}_A \\ -\boldsymbol{v}_B \end{bmatrix}.$$

◀

C Missing Details from Section 4

C.1 The Proof of Lemma 16

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. ◀

C.2 The Pseudocode of the Algorithm for Theorem 1

Let A_1 and A_2, two $\mathbb{F}$-weighted automata of total size s over pc monoid M with clique cover of size k given as input.

1. Following Theorem 13 construct the collection of algebraic branching programs $\{B_d\}_{d=1}^{n^3 s^2}$ over $\mathbb{F}\langle M \rangle$ of size sd and having d many layers.
2. For any d construct the branching program $B'_d = \psi(B_d)$ over $\mathbb{F}\langle \mathrm{X}'_1 \rangle \otimes \cdots \otimes \mathbb{F}\langle \mathrm{X}'_k \rangle$ of size at most $t_d = n^3 s^2 d$ and having kd many layers.
3. Following Lemma 14, construct the set $\mathcal{H}_{k,nk,kd,t_d} = \left(\mathcal{H}_{1,nk,kd,t_d(d+1)^{2(k-1)}}\right)^k$. Recall that the variables sets are $\{x_{ij}\}_{1 \leq i \leq n, 1 \leq j \leq k}$.
4. Now, for each $d \leq n^3 s^2$, do the following :
   a. For each point $(v_1, \ldots, v_k)$ in the set $\mathcal{H}_{k,nk,kd,t_d}$, substitute the variable x_{ij} by the matrix $v_j(x_{ij})$.
   b. Evaluate B'_d on such a substitution using matrix tensors, multiplications, and additions.
   c. If B'_d evaluates to a nonzero matrix, we stop the procedure and announce that A_1 and A_2 are not multiplicity equivalent.

5. Otherwise, for all $1 \leq d \leq n^3 s^2$, if B'_d evaluates to zero, declare that A_1 and A_2 are multiplicity equivalent.

C.3 The size upper bound for the ABP $\psi(B_d)$

Recall from Corollary 9 that $\psi(x_i) = f_1(x_i) \otimes f_2(x_i) \otimes \cdots \otimes f_k(x_i)$ which in turn is same as $(f_1(x_i) \otimes 1 \otimes \cdots \otimes 1) \cdot (1 \otimes f_2(x_i) \otimes \cdots \otimes 1) \cdots (1 \otimes 1 \otimes \cdots \otimes f_k(x_i))$. Given an ABP B_d computing $f \in \mathbb{F}\langle M \rangle$, we construct the algebraic branching program $\psi(B_d)$ by applying ψ to the edges of B_d and replacing each x_i by $\psi(x_i)$.

Let (p, q) be an edge in B_d labelled by the linear form $\sum_{i=1}^{n} \alpha_i x_i$. This can be thought of as n multi-edges each of which is labelled by $\alpha_i x_i$. Then each such edge (between p and q) which is labelled by $\alpha_i x_i$, is now replaced by a path $p = p_0, p_1, \ldots, p_k = q$ of length k where the edge (p_0, p_1) is labelled by $\alpha_i f_1(x_i) \otimes \cdots \otimes 1$ and the remaining edges (p_j, p_{j+1}) are labelled by $1 \otimes \cdots \otimes f_j(x_i) \otimes \cdots \otimes 1$ for $j \in [k-1]$. Clearly, $B'_d = \psi(B_d)$ computes a polynomial in $\mathbb{F}\langle \mathrm{X}'_1 \rangle \otimes \cdots \otimes \mathbb{F}\langle \mathrm{X}'_k \rangle$. Clearly, the size of B'_d is at most $t_d = nks^2 d$ and the number of layers is kd.

D Missing Details from Section 5

D.1 The Proof of Lemma 17

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 $f = \sum_{m \in \mathcal{M}_k} f_m \otimes m$ 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 non-zero 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 non-zero 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 A_k, such that $v_k(g)$ is a non-zero matrix of dimension $t_k(d)$. Hence, for the combined evaluation $\boldsymbol{v} = (\boldsymbol{v'}, v_k)$, $\boldsymbol{v}(f)$ is a non-zero matrix of dimension $\prod_{i=1}^{k} t_i(d)$. Using an union bound the failure probability can be bounded by $1/2$. ◀

D.2 Description of Transition Matrices of the Automaton in Figure 3

The transition matrices are the following:

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