Hadamard Product of Polynomials and the Identity Testing Problem

In this chapter we introduce and study the Hadamard product of the multivariate polynomials in the free noncommutative polynomial ring $\mathbb{F}\{x_1, x_2, \ldots , x_n\}$. We explore arithmetic circuit and branching program complexity of the Hadamard product of polynomials when they are individually given by arithmetic circuits and/or algebraic branching programs.

5.1 Introduction

Our definition of the Hadamard Product can be seen as an algebraic generalization of the intersection of the formal languages. The definition of Hadamard Product is motivated by the well known Hadamard product of matrices. Hadamard product of matrices of same dimension is simply entry-wise product. Next we define the Hadamard product of polynomials.

**Definition 5.1** Let $f, g \in \mathbb{F}\{X\}$ where $X = \{x_1, x_2, \ldots , x_n\}$. The Hadamard product of $f$ and $g$, denoted $f \circ g$, is the polynomial $f \circ g = \sum_m a_nb_n m$, where $f = \sum_m a_m m$ and $g = \sum_m b_m m$, where the sums index over monomials $m$. 
To see the connection of this definition with that of Hadamard product of two matrices we recall the definition of communication matrices \([N91]\) associated with a degree \(d\) homogeneous polynomial \(f \in \mathbb{F}\{X\}\). For \(k = 1, \ldots, d\) the communication matrix \(M_k(f)\) has its rows indexed by degree \(k\) monomials and columns by degree \(d - k\) monomials and the \((m, m')^{th}\) entry of \(M_k(f)\) is the coefficient of \(mm'\) in \(f\). It follows easily that Hadamard product of communication matrices associated with two polynomials \(f\) and \(g\) is same as the communication matrix associated with their Hadamard product (as defined above).

We show that the noncommutative branching program complexity of the Hadamard product \(f \circ g\) is upper bounded by the product of the branching program sizes for \(f\) and \(g\). This upper bound is natural because we know from Nisan’s seminal work \([N91]\) that the algebraic branching program (ABP) complexity \(B(f)\) is well characterized by the ranks of its “communication” matrices \(M_k(f)\), and the rank of Hadamard product \(A \circ B\) of two matrices \(A\) and \(B\) is upper bounded by the product of their ranks. Our proof is constructive: we give a deterministic logspace algorithm for computing an ABP for \(f \circ g\).

We then apply this result to tightly classify the identity testing problem for noncommutative ABPs over field of rationals. Before stating our main result we recall some complexity theory preliminaries.

We recall some definitions of logspace counting classes from \([AO96]\). Let \(L\) denote the class of languages accepted by deterministic logspace machines.

\(\text{GapL}\) is the class of functions \(f : \Sigma^* \rightarrow \mathbb{Z}\), for which there is a logspace bounded NDTM \(M\) such that for each input \(x \in \Sigma^*\), we have \(f(x) = \text{acc}_M(x) - \text{rej}_M(x)\), where \(\text{acc}_M(x)\) and \(\text{rej}_M(x)\) are the number of accepting and rejecting paths of \(M\) on input \(x\), respectively.

A language \(L\) is in \(C_wL\) if there exists a function \(f \in \text{GapL}\) such that \(x \in L\) if and only if \(f(x) = 0\). For a prime \(p\), a language \(L\) is in the complexity class \(\text{Mod}_pL\) if there exists a function \(f \in \text{GapL}\) such that \(x \in L\) if and only if \(f(x) = 0(\text{ mod } p)\).

It is shown in \([AO96]\) that checking if an integer matrix is singular is complete for \(C_wL\) with respect to logspace many-one reductions. The same problem is known to be
complete for Mod$_p$L over a field of characteristic $p$. It is useful to recall that both $C_=L$ and Mod$_p$L are contained in NC$^2$.

Main results in this chapter:

It is shown by Raz and Shpilka [RS05] that the polynomial identity testing problem for noncommutative ABPs can be solved in deterministic polynomial time. Using result on Hadamard Product of two ABPs, we show that the identity testing problem for non-commutative ABPs over rationals is equivalent to the matrix singularity problem under deterministic logspace many-one reductions. This implies the identity testing problem in case of rationals is $C_=L$-complete. In particular it gives NC$^2$ upperbound on the complexity of the problem.

We show that, the identity testing problem for the noncommutative ABPs over finite field of characteristic $p$ is equivalent to Matrix Singularity problem over field of characteristic $p$ under randomized logspace reduction. Firstly this reduction shows a randomized NC$^2$ upperbound on the complexity of the problem and it follows from [AO96] that the problem is in randomized Mod$_p$L. Using standard amplification techniques we get a Mod$_p$L/Poly upper bound. We also investigate the parallel complexity of the problem. We show that Raz-Shpilka identity test can be parallelized which gives a NC$^3$ upper-bound for the identity testing problem for the non-commutative ABPs over any field.

It turns out that the problem is hard (with respect to logspace many-one reductions) for both NL and Mod$_p$L. Hence, it is not likely to be easy to improve the upper bound unconditionally to Mod$_p$L (it would imply that NL is contained in Mod$_p$L). However, under a hardness assumption we can apply standard arguments [ARZ99, KvM02] to derandomize this algorithm and put the problem in Mod$_p$L.

It is an interesting question whether we can show deterministic NC$^2$ upper bound on the identity testing problem for noncommutative ABPs over finite fields?

Next explore the expressive power of the Hadamard product of two polynomials when either or both of them given by arithmetic circuit. We show that if either of the two polynomials is given by an ABP the we can efficiently (in logspace) compute an arithmetic circuit for the Hadamard product of the polynomials. But if both the polynomials are give by arithmetic circuits then it is not easy to come up with an efficient algorithm
to compute an arithmetic circuit for the Hadamard product of the two polynomials (We show that such an algorithm would imply a non-trivial circuit-size lowerbound).

We also consider following identity testing question: Given two polynomials \( f, g \in \mathbb{F}\{X\} \) either by an ABP or by an arithmetic circuit check whether \( f \circ g \) is identically zero. We show that if both the polynomials are given by arithmetic circuits then the problem is coNP-hard even when the circuits are monotone. Whereas, if either of the polynomials is given by an ABP the problem has polynomial time algorithm.

## 5.2 The Hadamard Product

Let \( f, g \in \mathbb{F}\{X\} \) where \( X = \{x_1, x_2, \ldots, x_n\} \). Clearly, \( \text{mon}(f \circ g) = \text{mon}(f) \cap \text{mon}(g) \).

Thus, the Hadamard product can be seen as an algebraic version of the intersection of formal languages. Our definition of the Hadamard product of polynomials is actually motivated by the well-known Hadamard product \( A \circ B \) of two \( m \times n \) matrices \( A \) and \( B \).

We recall the following well-known bound for the rank of the Hadamard product.

**Proposition 5.2** Let \( A \) and \( B \) be \( m \times n \) matrices over a field \( \mathbb{F} \). Then \( \text{rank}(A \circ B) \leq \text{rank}(A)\text{rank}(B) \).

It is known from Nisan’s work [N91] that the ABP complexity \( B(f) \) of a polynomial \( f \in \mathbb{F}\{X\} \) is closely connected with the ranks of the communication matrices \( M_k(f) \), where \( M_k(f) \) has its rows indexed by degree \( k \) monomials and columns by degree \( d - k \) monomials and the \((m, m')^\text{th}\) entry of \( M_k(f) \) is the coefficient of \( mm' \) in \( f \). Nisan showed that \( B(f) = \sum_k \text{rank}(M_k(f)) \). Nisan’s result and the above proposition easily imply the following bound on the ABP complexity of \( f \circ g \).

**Lemma 5.3** For \( f, g \in \mathbb{F}\{X\} \) we have \( B(f \circ g) \leq B(f)B(g) \).

**Proof** By Nisan’s result \( B(f \circ g) = \sum_k \text{rank}(M_k(f \circ g)) \). The above proposition implies

\[
\sum_k \text{rank}(M_k(f \circ g)) \leq \left( \sum_k \text{rank}(M_k(f)) \text{rank}(M_k(g)) \right) \leq \left( \sum_k \text{rank}(M_k(f)) \right) \left( \sum_k \text{rank}(M_k(g)) \right),
\]
Chapter 5. Hadamard Product of Polynomials and the Identity Testing Problem

and the claim follows.

We now show an algorithmic version of this upper bound.

**Theorem 5.4** Let $P$ and $Q$ be two given ABP’s computing polynomials $f$ and $g$ in $\mathbb{F}\{x_1, x_2, \ldots, x_n\}$, respectively. Then there is a deterministic polynomial-time algorithm that will output an ABP $R$ for the polynomial $f \circ g$ such that the size of $R$ is a constant multiple of the product of the sizes of $P$ and $Q$. (Indeed, $R$ can be computed in deterministic logspace.)

**Proof** Let $f_i$ and $g_i$ denote the $i^{th}$ homogeneous parts of $f$ and $g$ respectively. Then $f = \sum_{i=0}^{d} f_i$ and $g = \sum_{i=0}^{d} g_i$. Since the Hadamard product is distributive over addition and $f_i \circ g_j = 0$ for $i \neq j$ we have $f \circ g = \sum_{i=0}^{d} f_i \circ g_i$. Thus, we can assume that both $P$ and $Q$ are homogeneous ABP’s of degree $d$. Otherwise, we can easily construct an ABP to compute $f_i \circ g_i$ separately for each $i$ and put them together. Note that we can easily compute ABPs for $f_i$ and $g_i$ in logspace given as an input the ABPs for $f$ and $g$.

By allowing parallel edges between nodes of $P$ and $Q$ we can assume that the labels associated with each edge in an ABP is either 0 or $\alpha x_i$ for some variable $x_i$ and scalar $\alpha \in \mathbb{F}$. Let $s_1$ and $s_2$ bound the number of nodes in each layer of $P$ and $Q$ respectively. Denote the $j^{th}$ node in layer $i$ by $\langle i, j \rangle$ for ABPs $P$ and $Q$. Now we describe the construction of the ABP $R$ for computing the polynomial $f \circ g$. Each layer $i$, $1 \leq i \leq d$ of $R$ will have $s_1 \cdot s_2$ nodes, with node labeled $\langle i, a, b \rangle$ corresponding to the node $\langle i, a \rangle$ of $P$ and the node $\langle i, b \rangle$ of $Q$. We can assume that there is an edge from the every node in the layer $i$ to the every node in the layer $i+1$ for both ABPs. If there is no such edge we can always include it with label 0.

In the new ABP $R$ we put an edge from $\langle i, a, b \rangle$ to $\langle i + 1, c, e \rangle$ with label $\alpha \beta x_t$ if and only if there is an edge from node $\langle i, a \rangle$ to $\langle i + 1, c \rangle$ with label $\alpha x_t$ in $P$ and an edge from $\langle i, b \rangle$ to $\langle i + 1, e \rangle$ with label $\beta x_t$ in ABP $Q$. Let $\langle 0, a, b \rangle$ and $\langle d, c, e \rangle$ denote the source and the sink nodes of ABP $R$, where $\langle 0, a \rangle, \langle 0, b \rangle$ are the source nodes of $P$ and $Q$, and $\langle d, c \rangle, \langle d, e \rangle$ are the sink nodes of $P$ and $Q$ respectively. It is easy to see that ABP $R$ can be computed in deterministic logspace. Let $h_{\langle i, a, b \rangle}$ denote the polynomial computed at node $\langle i, a, b \rangle$ of ABP $R$. Similarly, let $f_{\langle i, a \rangle}$ and $g_{\langle i, b \rangle}$ denote the polynomials computed
Chapter 5. Hadamard Product of Polynomials and the Identity Testing Problem

at node \( \langle i, a \rangle \) of \( P \) and node \( \langle i, b \rangle \) of \( Q \). We can easily check that 
\[
h_{\langle i,a,b \rangle} = f_{\langle i,a \rangle} \circ g_{\langle i,b \rangle}
\]
by an induction argument on the number of layers in the ABPs. It follows from this
inductive argument that the ABP \( R \) computes the polynomial \( f \circ g \) at its sink node. The
bound on the size of \( R \) also follows easily. 

\[ \hfill \]

5.3 Identity Testing for noncommutative ABPs

In this section we explore the complexity of the polynomial identity testing problem for
noncommutative ABPs. First we show that applying the above theorem we can get a
tight complexity theoretic upper bound for the identity testing problem for noncommu-
tative ABPs over rationals.

**Theorem 5.5** The problem of polynomial identity testing for noncommutative algebraic
branching programs over \( \mathbb{Q} \) is in \( \text{NC}^2 \). More precisely, it complete for the logspace
counting class \( \text{C}_L \) under logspace reductions.

**Proof** Let \( P \) be the given ABP computing \( f \in \mathbb{Q}\{X\} \). We apply the construction of
the Theorem 5.4 to compute a polynomial sized ABP \( R \) for the Hadamard product \( f \circ f \).
Notice that \( f \circ f \) is nonzero iff \( f \) is nonzero. Now, we crucially use the fact that \( f \circ f \)
is a polynomial whose nonzero coefficients are all positive. Hence, \( f \circ f \) is nonzero iff
it evaluates to a nonzero number on the all \( 1 \)'s input. The problem thus boils down to
checking if \( R \) evaluates to a nonzero number on the all \( 1 \)'s input.

By Theorem 5.4, the ABP \( R \) for polynomial \( f \circ f \) is computable in deterministic
logspace, given as input an ABP for \( f \). Furthermore, evaluating the ABP \( R \) on the
all \( 1 \)'s input can be easily converted to iterated integer matrix multiplication (one matrix
for each layer of the ABP), and checking if \( R \) evaluates to a nonzero number can be
done by checking if a specific entry of the product matrix is nonzero. It is well known
that checking if a specific entry of an iterated integer matrix product is zero is in the
logspace counting class \( \text{C}_L \) (e.g. see [AO96, ABO99]). However, \( \text{C}_L \) is contained in
\( \text{NC}^2 \).

We now argue that the problem is hard for \( \text{C}_L \). The problem of checking if an integer
matrix \( A \) is singular is well known to be complete for \( \text{C}_L \) under deterministic logspace
reductions. The standard GapL algorithm for computing \( \det(A) \) [T91] can be converted to an ABP \( P_A \) which will compute \( \det(A) \). Hence the ABP \( P_A \) computes the identically zero polynomial iff \( A \) is singular. Putting it all together, it follows that identity testing of noncommutative ABPs over rationals is complete for the class \( C_{\L} \).

**An iterative matrix product problem** Suppose \( B \) is a noncommutative ABP computing a homogeneous polynomial in \( \mathbb{F}\{X\} \) of degree \( d \), where each edge of the ABP is labeled by a homogeneous linear form in variables from \( X \).

Let \( n_\ell \) denote the number of nodes of \( B \) in layer \( \ell \), \( 0 \leq \ell \leq d \). For each \( x_i \) and layer \( \ell \), we associate an \( n_\ell \times n_{\ell + 1} \) matrix \( A_{i,\ell} \) where the \((k,j)\)th entry of matrix \( A_{i,\ell} \) is the coefficient of \( x_i \) in the linear form associated with the \((v_k,u_j)\) edge in the ABP \( B \). Here \( v_k \) is the \( k^{th} \) node in layer \( \ell \) and \( u_j \) the \( j^{th} \) node in the layer \( \ell + 1 \). The following claim is easy to see and relates these matrices to the ABP \( B \).

**Claim 8** The coefficient of any degree \( d \) monomial \( x_{i_1}x_{i_2}\cdots x_{i_d} \) in the polynomial computed by the ABP \( B \) is the matrix product \( A_{i_1,0}A_{i_2,1}\cdots A_{i_d,d-1} \) (which is a scalar since \( A_{i_1,0} \) is a row and \( A_{i_d,d-1} \) is a column).

Let \( i \) and \( j \) be any two nodes in the ABP \( B \). We denote by \( B(i,j) \) the algebraic branching program obtained from the ABP \( B \) by designating node \( i \) in \( B \) as the source node and node \( j \) as the sink node. Clearly, \( B(i,j) \) computes a homogeneous polynomial of degree \( b - a \) if \( i \) appears in layer \( a \) and \( j \) in layer \( b \).

For layers \( a, b \), \( 0 \leq a < b \leq d \) let \( t = b - a \) and \( P(a,b) = \{ A_{s_1,a}A_{s_2,a+1}\cdots A_{s_t,b-1} \mid 1 \leq s_j \leq n, \text{ for } 1 \leq j \leq t \} \). \( P(a,b) \) consists of \( n_a \times n_b \) matrices. Thus the dimension of the linear space spanned by \( P(a,b) \) is bounded by \( n_a n_b \). It follows from Claim 8 that the linear span of \( P(a,b) \) is the zero space iff the polynomial computed by ABP \( B(i,j) \) is identically zero for every \( 1 \leq i \leq n_a \) and \( 1 \leq j \leq n_b \).

Thus, it suffices to compute a basis for the space spanned by matrices in \( P(0,d) \) to check whether the polynomial computed by \( B \) is identically zero. We can easily give a deterministic NC\(^3\) algorithm for this problem over any field \( \mathbb{F} \): First recursively compute bases \( M_1 \) and \( M_2 \) for the space spanned by matrices in \( P(0,d/2) \) and \( P(d/2+1,d) \) respectively. From bases \( M_1 \) and \( M_2 \) we can compute in deterministic NC\(^2\) a basis \( M \)
for space spanned by matrices in $P(0, d)$ as follows. We compute the set $S$ of pairwise products of matrices in $M_1$ and $M_2$ and then we can compute a maximal linearly independent subset of $S$ in NC$^2$ (see e.g. [ABO99]). This gives an easy NC$^3$ algorithm to compute a basis for the linear span of $P(0, d)$. This proves the following.

**Theorem 5.6** The problem of polynomial identity testing for noncommutative algebraic branching programs over any field (in particular, finite fields $\mathbb{F}$) is in deterministic NC$^3$.

Can we give a tight complexity characterization for identity testing of noncommutative ABPs over finite fields? We show that the problem is in nonuniform Mod$^p_L$ and is hard for Mod$^p_L$ under logspace reductions. Furthermore, the problem is hard for NL. Hence, it appears difficult to improve the upper bound to uniform Mod$^p_L$ (as NL is not known to be contained in uniform Mod$^p_L$).

**Theorem 5.7** The problem of polynomial identity testing for noncommutative algebraic branching programs over a finite field $\mathbb{F}$ of characteristic $p$ is in Mod$^p_L$/Poly.

**Proof** Consider a new ABP $B'$ in which we replace the variables $x_i$, $1 \leq i \leq n$ appearing in the linear form associated with an edge from some node in the layer $l$ to a node in the layer $l + 1$ of ABP $B$ by new variable $x_{i,l}$, for layers $l = 0, 1, \ldots, d - 1$. Let $g \in \mathbb{F}[X]$ denotes the polynomial computed by the ABP $B'$ in commuting variables $x_{i,l}$, $1 \leq i \leq n$, $1 \leq l < d$. It is easy to see that the commutative polynomial $g \in \mathbb{F}[X]$ is identically zero iff the noncommutative polynomial $f \in \mathbb{F}\{X\}$ computed by ABP $B$ is identically zero. Now, we can apply the standard Schwartz-Zippel lemma to check if $g$ is identically zero by substituting random values for the variables $x_{i,l}$ from $\mathbb{F}$ (or a suitable finite extension of $\mathbb{F}$). After substitution of field elements, we are left with an iterated matrix product over a field of characteristic $p$ which can be done in Mod$^p_L$. This gives us a randomized Mod$^p_L$ algorithm. By standard amplification it follows that the problem is in Mod$^p_L$/Poly.

Next we show that identity testing problem for noncommutative ABPs over any field is hard for NL by a reduction from directed graph reachability. Let $(G, s, t)$ be a reachability instance. Without loss of generality, we assume that $G$ is a layered directed acyclic
Theorem 5.8 The problem of polynomial identity testing for noncommutative algebraic branching programs over any field is hard for NL.

5.4 Hadamard product of noncommutative circuits

In this section we study the expressive power of Hadamard product of two polynomial when one or both of them are given by an arithmetic circuits rather than an ABP. Analogous to Theorem 5.4 we show that \( f \circ g \) has small circuits if \( f \) has a small circuit and \( g \) has a small ABP.

Theorem 5.9 Let \( f, h \in \mathbb{F}\{x_1, x_2, \ldots , x_n\} \) be given by a degree \( d \) circuit \( C \) and a degree \( d \) ABP \( P \) respectively, where \( d = O(n^{O(1)}) \). Then we can compute in polynomial time a circuit \( C' \) that computes \( f \circ h \) where the size of \( C' \) is polynomially bounded in the sizes of \( C \) and \( P \).

Proof As in the proof of Theorem 5.4 we can assume that both \( f \) and \( h \) are homogeneous polynomials of degree \( d \). Let \( f_g \) denote the polynomial computed at gate \( g \) of circuit \( C \). Let \( w \) bound the number of nodes in any layer of \( P \). Let \( \langle i, a \rangle \) denote the \( a^{th} \) node in the \( i^{th} \) layer of \( P \) for \( 0 \leq i \leq d, 1 \leq a \leq w \). Let \( h_{\langle i, a \rangle, \langle j, b \rangle} \) denote the polynomial computed by ABP \( P' \), where \( P' \) is same as \( P \) but with source node \( \langle i, a \rangle \) and sink node \( \langle j, b \rangle \). We now describe the circuit \( C' \) computing the polynomial \( f \circ h \). In \( C' \) we have gates \( \langle g, l, (i, a), (i+l, b) \rangle \) for \( 0 \leq l \leq d, 0 \leq i \leq d, 1 \leq a, b \leq w \) associated with each gate \( g \) of \( C \), such that at the gate \( \langle g, l, (i, a), (i+l, b) \rangle \) the circuit \( C' \) computes

\[
r_{\langle g, l \rangle, (i+l, b)} = f_{\langle g, l \rangle} \circ h_{\langle i, a \rangle, (i+l, b)}
\]

where \( f_{\langle g, l \rangle} \) denotes the degree \( l \) homogeneous component of the polynomial \( f_g \).
Chapter 5. Hadamard Product of Polynomials and the Identity Testing Problem

If \( g \) is a \(+\) gate of \( C \) with input gates \( g_1, g_2 \) so that \( f_g = f_{g_1} + f_{g_2} \), we have 
\[
\rho_{i,a,(i+l,b)}^{g,f} = \rho_{i,a,(i+l,b)}^{g_1,f} + \rho_{i,a,(i+l,b)}^{g_2,f}, \text{ for } 0 \leq l \leq d, 0 \leq i \leq d, 1 \leq a, b \leq w. \]
In other words, \( \langle g, l, (i, a), (i + l, b) \rangle \) is a \(+\) gate in \( C' \) with input gates \( \langle g_1, l, (i, a), (i + l, b) \rangle \) and \( \langle g_2, l, (i, a), (i + l, b) \rangle \). If \( f \) is a \( \times \) gate in \( C \) we will have
\[
\rho_{i,a,(i+l,b)}^{g,f} = \sum_{j=0}^{l} \sum_{t=1}^{w} \rho_{i,a,(i+j,t)}^{g_1,f} \cdot \rho_{i+j,t,(i+l,b)}^{g_2,f} \tag{5.2}
\]

The above formula is easily computable by a small subcircuit. The output gate of \( C' \) will be \( \langle g, d, (0,1), (d,1) \rangle \), where \( g \) is the output gate of \( C \), and \( (0,1) \) and \( (d,1) \) are the source and the sink of the ABP \( P \) respectively. This is the description of the circuit \( C' \).

We inductively argue that gate \( \langle g, l, (i, a), (i + l, b) \rangle \) of \( C' \) computes the polynomial \( f_{g,l} \circ h_{i,a,(i+l,b)} \). If \( g \) is a \(+\) gate of \( C \) the claim is obvious. Suppose \( g \) is a \( \times \) gate of \( C \) with inputs \( g_1, g_2 \) such that \( f_g = f_{g_1} \cdot f_{g_2} \). Inductively assume that the claim holds for the gates \( g_1 \) and \( g_2 \). Then we have 
\[
f_{g,l} \circ h_{i,a,(i+l,b)} = \sum_{i=0}^{l} f_{g_1,l} \cdot f_{g_2,l-i} \cdot h_{i,a,(i+l,b)}
\]

By induction hypothesis we have 
\[
\rho_{i,a,(i+l,t)}^{g_1,f} = \rho_{i,a,(i+l,t)}^{g_1,f} \circ h_{i,a,(i+l,t)} \text{ and } \rho_{i,a,(i+l,t)}^{g_2,p} = \rho_{i,a,(i+l,t)}^{g_2,p} \circ h_{i,a,(i+l,t)}.
\]

Now, from Equation 5.2 it is easy to obtain the desired Equation 5.1. Therefore, at the output gate \( \langle g, d, (0,1), (d,1) \rangle \) the circuit \( C' \) computes \( f \circ h \).

The size of \( C' \) is bounded by a polynomial in the sizes of \( C \) and \( P \). □

On the other hand, suppose \( f \) and \( g \) individually have small circuit complexity. Does \( f \circ g \) have small circuit complexity? Can we compute such a circuit for \( f \circ g \) from circuits for \( f \) and \( g \)? We first consider these questions for monotone circuits. It is useful to understand the connection between monotone noncommutative circuits and context-
Chapter 5. Hadamard Product of Polynomials and the Identity Testing Problem

free grammars.

**Theorem 5.10** There are monotone circuits $C$ and $C'$ computing polynomials $f$ and $g$ in $\mathbb{Q}\{X\}$ respectively, such that the polynomial $f \circ g$ requires monotone circuits of size exponential in $|X|$, size$(C)$, and size$(C')$.

**Proof** Let $X = \{x_1, \ldots, x_n\}$. Define the finite language $L_1 = \{zww^r \mid z, w \in X^*, |z| = |w| = n\}$ and the corresponding polynomial $f = \sum_{m_\alpha \in L_1} m_\alpha$. Similarly let $L_2 = \{ww^rz \mid z, w \in X^*, |z| = |w| = n\}$, and the corresponding polynomial $g = \sum_{m_\alpha \in L_2} m_\alpha$. It is easy to see that there are $\text{Poly}(n)$ size unambiguous acyclic CFGs for $L_1$ and $L_2$. Hence, by Proposition 2.134.13 there are monotone circuits $C_1$ and $C_2$ of size Poly$(n)$ such that $C_1$ computes polynomial $f$ and $C_2$ computes polynomial $g$. We first show that the finite language $L_1 \cap L_2$ cannot be generated by any acyclic CFG of size $2^{o(n \log n)}$. Assume to the contrary that there is an acyclic CFG $G = (V, T, P, S)$ for $L_1 \cap L_2$ of size $2^{o(n \log n)}$. Notice that $L_1 \cap L_2 = \{t \mid t = ww^rw, w \in X^*, |w| = n\}$.

Consider any derivation tree $T'$ for a word $ww^rw = w_1w_2 \ldots w_nw_nw_{n-1} \ldots w_2w_1w_1 \ldots w_n$. Starting from the root of the binary tree $T'$, we traverse down the tree always picking the child with larger yield. Clearly, there must be a nonterminal $A \in V$ in this path of the derivation tree such that $A \Rightarrow^* u, u \in X^*$ and $n \leq |u| < 2n$. Crucially, note that any word that $A$ generates must have same length since every word generated by the grammar $G$ is in $L_1 \cap L_2$ and hence of length $3n$. Let $ww^rw = s_1us_2$ where $|s_1| = k$. As $|u| < 2n$, the string $s_1s_2$ completely determines the string $ww^rw$. Hence, the nonterminal $A$ can derive at most one string $u$. Furthermore, this string $u$ can occur in at most $2n$ positions in a string of length $3n$. Notice that for each position in which $u$ can occur it completely determines a string of the form $ww^rw$. Therefore, $A$ can participate in the derivation of at most $2n$ strings from $L_1 \cap L_2$. Since there are $n^2$ distinct words in $L_1 \cap L_2$, it follows that there must be at least $\frac{n^3}{2n}$ distinct nonterminals in $V$. This contradicts the size assumption of $G$.

Since $L_1 \cap L_2$ cannot be generated by any acyclic CFG of size $2^{o(n \log n)}$, it follows that the polynomial $f \circ g$ can not be computed by any monotone circuit of $2^{o(n \log n)}$ size. \qed
Theorem 5.10 shows that the Hadamard product of monotone circuits is more expressive than monotone circuits. It raises the question whether the permanent polynomial can be expressed as the Hadamard product of polynomial-size (or even subexponential size) monotone circuits. We note here that the permanent can be easily expressed as the Hadamard product of $O(n^3)$ many monotone circuits (in fact, monotone ABPs).

**Theorem 5.11** Suppose there is a deterministic subexponential-time algorithm that takes two circuits as input, computing polynomials $f$ and $g$ in $\mathbb{Q}\{x_1, \ldots, x_n\}$, and outputs a circuit for $f \circ g$. Then either NEXP is not in $P$/Poly or the Permanent does not have polynomial size noncommutative circuits.

**Proof** Let $C_1$ be a circuit computing some polynomial $h \in \mathbb{Q}\{x_1, \ldots, x_n\}$. By assumption, we can compute a circuit $C_2$ for $h \circ h$ in subexponential time. Therefore, $h$ is identically zero iff $h \circ h$ is identically zero iff $C_2$ evaluates to 0 on the all 1’s input. We can easily check if $C_2$ evaluates to 0 on all 1’s input by substitution and evaluation. This gives a deterministic subexponential time algorithm for testing if $h$ is identically zero. By the noncommutative analogue of [KI03], shown in [AMS08], it follows that either NEXP $\not\subset P$/Poly or the Permanent does not have polynomial size noncommutative circuits. □

Next, we show that the identity testing problem: given $f, g \in \mathbb{F}\{X\}$ by circuits test if $f \circ g$ is identically zero is coNP hard.

**Theorem 5.12** Given two monotone polynomial-degree circuits $C$ and $C'$ computing polynomial $f, g \in \mathbb{Q}\{X\}$ it is coNP-complete to check if $f \circ g$ is identically zero.

**Proof** We first show that the complement of the problem is in NP. The NP machine will guess a monomial $m_\alpha \in X^*$, $X = \{x_1, \ldots, x_n\}$ and check if coefficient of $m_\alpha$ is nonzero in both $C$ and $C'$. Note that we can compute the coefficient of $m_\alpha$ in $C$ and $C'$ in deterministic polynomial time [AMS08]. Denote by CFGINT the problem of testing nonemptiness of the intersection of two acyclic CFGs that generate Poly($n$) length strings. By Lemma 4.13 CFGINT is polynomial time many-one reducible to testing if $f \circ g$ is identically zero. The problem of testing if the intersection of two CFGs (with
recursion) is nonempty is known to be undecidable via a reduction from the Post Correspondence problem [HMU, Chapter 9, Page 422]. We can give an analogous reduction from bounded Post Correspondence to CFGINT. The NP-hardness of CFGINT follows from the NP-hardness of bounded Post Correspondence [GJ79].

5.5 Overview

In this chapter we introduced and studied the Hadamard product of the multivariate polynomials in the free noncommutative polynomial ring $F\{x_1, x_2, \ldots, x_n\}$. We explored arithmetic circuit and branching program complexity of the Hadamard product of polynomials when they are individually given by arithmetic circuits and/or algebraic branching programs.

When both of the polynomials are given by algebraic branching programs we gave a logspace algorithm to generate an ABP which computes the Hadamard product of the two polynomials. Using this result we showed that the identity testing problems for noncommutative ABPs over rationals is complete for the logspace counting class $C_L$ (which is known to be contained in $NC^2$). We have slightly weaker results for the identity testing problem for noncommutative ABPs over finite fields. We gave Mod$_p$L/Poly and $NC^3$ upperbound on the complexity of the problem, where $p$ is the characteristic of the field. It easily follows that the concerned identity testing problem is hard for NL. So it is difficult to improve above bound unconditionally to Mod$_p$L (as it would show NL $\subseteq$ Mod$_p$L which is an open problem). An interesting question in this context is: can we show deterministic NC$^2$ upperbound for the identity testing problem over noncommutative ABPs over finite fields?

We also explored the expressive power of the Hadamard product of two polynomials when either or both of them given by arithmetic circuit. We showed that if either of the two polynomials is given by an ABP the we can efficiently (in logspace) compute an arithmetic circuit for the Hadamard product of the polynomials. But if both the polynomials are given by arithmetic circuits then it is not easy to come up with an efficient algorithm to compute an arithmetic circuit for the Hadamard product of the two polynomials (such an algorithm would imply a non-trivial circuit-size lowerbound).
Other question we studied is the following identity testing problem: Given two polynomials \( f, g \in \mathbb{F}\{X\} \) either by an ABP or by an arithmetic circuit check whether \( f \circ g \) is identically zero. We showed that if both the polynomials are given by arithmetic circuits then the problem is coNP-hard even when the circuits are monotone. Whereas, if either of the polynomials is given by an ABP the problem has polynomial time algorithm.