Dynamic Parallel Complexity of Computational Circuits*

Gary L.Miller Shang-Hua Teng

Department of Computer Science
University of Southern California
Los Angeles, California 90089

Abstract

The dynamic parallel complexity of general computational circuits (defined in introduction) is discussed. We exhibit some relationships between parallel circuit evaluation and some uniform closure properties of a certain class of unary functions and present a systematic method for the design of processor efficient parallel algorithms for circuit evaluation. Using this method: (1) we improve the algorithm for parallel Boolean circuit evaluation;(2) we give a nontrivial upper bound for parallel min-max-plus circuit evaluation; (3) we partially answer the first open question raised in [MiRK85] by showing that all circuits over finite noncommutative semi-ring and circuits over infinite noncommutative semi-ring which has finite dimension over a commutative semi-ring can be evaluated in polylogarithmic time in its size and degree using $M(n)$ processors. Moreover, we develop a theory for determining closure properties of certain classes of unary functions.

1 Introduction

The problems of parallel arithmetic circuit (polynomial, straight-line program) evaluation was intensively studied. Valiant, Skyum, Berkowitz, Rackoff [VSBR83] proved that if a polynomial over $\{+,-,\times\}$ can be evaluated in C time sequentially, then with preprocessing (off-line), it can be evaluated in $O((\log d)(\log C + \log dn))$ times using only $C^{O(1)}$ processors in parallel. Miller, Ramachandran, Kaltofen [MiRK85] improved this result and showed that any arithmetic circuit with size n over a commutative semi-ring can evaluated in $O((\log n)(\log nd))$ times with $M(n)$ processors[1] on-line. However, there is no theory for parallel circuit evaluation when the circuit is not over a commutative semi-ring. Moreover, the previous results did not give a tight bound for parallel Boolean circuit evaluation.

A general CE-circuit is a $4-tuple$ $\{\mathcal{D}, \mathcal{OP}, \mathcal{F}, C\}$, where C is a labeled directed acyclic simple graph whose leaf nodes (the nodes with zero in-degree) are labeled with values from certain domain $\mathcal{D}$, whose internal nodes are labeled with operators from a set of operators $\mathcal{OP}$ and whose edges are labeled with unary functions from $\mathcal{F}$. The operators in $\mathcal{OP}$ are partitioned into *cheap operators* and *expensive operators*. The degree of a circuit is defined by mapping *cheap operators* to addition and mapping *expensive operators* to multiplication and then compute the formal degree [MiRK85] of the corresponding arithmetic circuit. The values of the nodes in a circuit are defined in the usual way. Our task is to compute the values of all nodes.

In this paper, we present a systematic method via the uniform closure properties of certain classes of unary functions for the design of processor efficient parallel algorithms for circuit evaluation in polylogarithmic time in its size and degree. We also develop a theory for proving closure properties of certain classes of unary functions. (see reduction lemma, section 6). The ideas in this paper can be used for systematic tree based parallel algorithm development [MiTe85]. Here are some major results we shall present in this paper.

- **Parallel Boolean Circuit Evaluation:** A Boolean circuit over a Boolean algebra $\mathcal{B} = \{\mathcal{D}, \vee, \wedge\}$ with unary function class $\mathcal{F} = \{(a \wedge x) \vee b \mid a, b \in \mathcal{D}\}$ can be evaluated in time no more than $O((\log n)(\log(min(d_{\wedge\vee}, d_{\vee\wedge})))$ using $M(n)$ processors. Moreover there are Boolean circuits with exponential $d_{\wedge\vee}$ and $d_{\vee\wedge}$ which can be evaluated in polylogarithmic time using $M(n)$ processors, without knowing the structure of the circuit beforehand. Where $d_{\wedge\vee}$ ($d_{\vee\wedge}$) is the degree of the Boolean cir-

*This work is supported by National Science Foundation grant DCR-8514961

[1] $M(n)$ is the number of processors required for multiplying $n \times n$ matrices over the semi-ring in $O(\log n)$ time.

cuit when we view $\vee$ ($\wedge$) as addition in the semi-ring formed by $\mathcal{B}$.

- **Parallel Min-Max-Plus Circuit Evaluation:** Any **min-max-plus** circuit C with n nodes, m max nodes and degree d can be evaluated in $O((\log n)(logdn)))$ time using $O(m \cdot M(n))$ processors.
- **Parallel Noncommutative Ring Circuit Evaluation:** We show our methods can be applied to certain circuit value problems over noncommutative semi-rings. We show that all circuits over a finite noncommutative semi-ring can be evaluated in polylogarithmic time in its size and degree using $M(n)$ processors. We also give a more processor efficient method to handle matrix rings over a commutative base ring. The naive approach would be to expand the matrix products and additions into operations over the base ring.

2 The Roles of Unary Functions in Parallel Circuit Evaluation

For easier understanding, we make three restrictions to the computational circuit in this detailed abstract. The theory about general computational circuit is given in the full paper. The first restriction is that there is only one cheap operator which is associative, and commutative, and the expensive operations distribute over it. The next two restrictions will be removed in the case of Boolean circuits. We can also drop these restrictions by using the method presented in section 5. The second restriction is that each *expensive node* has in-degree two. The third restriction is that there is no edge from *expensive nodes* to *expensive nodes*. Using these restrictions we define the value of a circuit:

Definition 2.1 (Value of nodes in circuit) *The value of a node in a circuit (denoted by $\underline{value}(v)$) is defined inductively:*

- *If v is a leaf, then $\underline{value}(v)$ is the value associated with v.*
- *If v is a cheap node, and labeled by $\oplus$, then $\underline{value}(v)$ $= \oplus(f_1(\underline{value}(v_1)), \ldots, f_k(\underline{value}(v_k)))$ where $v_1, \ldots, v_k$ are the children of v. $f_1, \ldots, f_k$ are the unary functions with edges $(v_1, v), \ldots, (v_k, v)$.*
- *If v is an expensive node, and labeled by $\otimes$, then: $\underline{value}(v) = f_1(v_1) \otimes f_2(v_2)$ where v_1, v_2 are the children of v. f_1, f_2 are unary functions with edges $(v_1, v), (v_2, v)$.*

2.1 Closure Properties

Definition 2.2 (Composition Closure Property) *A class of unary functions $\mathcal{F}$ is closed under composition if for all functions $f_1(x), f_2(x) \in \mathcal{F}$, $f_2(f_1(x)) \in \mathcal{F}$. We denote $f_2(f_1(x))$ by $f_2 \circ f_1(x)$.*

Definition 2.3 (Combination Closure Property) *A class of unary functions $\mathcal{F}$ over domain D is closed under combination over a binary, associative, commutative operator $\oplus$ if for all $f, g \in \mathcal{F}$, $f(x) \oplus g(x) \in \mathcal{F}$. (denoted by $f \oplus g$).*

Definition 2.4 (Projection Closure Property) *A class of unary functions $\mathcal{F}$ over domain D is closed under projection over a set of binary operators OP if for all $f \in \mathcal{F}, \odot \in OP, a \in D$ $f(x \odot a) \in \mathcal{F}$ and $f(a \odot x) \in \mathcal{F}$.*

Definition 2.5 (Linear Property[3]) *Let $\oplus$ be a binary operator which is associative and commutative. A class of unary functions $\mathcal{F}$ over domain D is linear[2] over $\oplus$ if for all $f \in \mathcal{F}, x, y \in D$, $f(x \oplus y) = f(x) \oplus f(y)$. Sometimes we say $\mathcal{F}$ distributes over $\oplus$.*

Claim 2.1 *If $\oplus$ is the cheap operation and $\mathcal{F}$ is a set of unary functions which distributes over $\oplus$ then the closure of $\mathcal{F}$ under composition, combination over $\oplus$, and projection over the expensive operations will still distribute over $\oplus$.*

Definition 2.6 *A class of unary functions $\mathcal{F}$ is closed over the pair of operation sets (OP_1, OP_2) if $\mathcal{F}$ is closed under composition, combination over all operators in OP_1, linear over OP_1, projection over OP_2.*

Claim 2.2 *$\{\mathcal{F}, \oplus, \circ\}$ forms a semi-ring.*

In later section, we use M(n) denotes the number of processors required for multiplying $n \times n$ matrices over semi-ring $\{\mathcal{F}, \oplus, \circ\}$ in $O(\log n)$ time.

2.2 Operations and Algorithm

We now define some notations which will make the specification of our parallel computational circuit concise.

$$U(C,C)_{ij} = \begin{cases} f_{ij} & \textit{if } v_i \textit{ and } v_j \textit{ are cheap} \\ 0 & \textit{otherwise} \end{cases}$$
$$U(X,C)_{ij} = \begin{cases} f_{ij} & \textit{if } v_j \textit{ is cheap} \\ 0 & \textit{otherwise} \end{cases}$$
$$U(X,X)_{ij} = \begin{cases} f_{ij} & \textit{if } v_i \textit{ or } v_j \textit{ is not cheap} \\ 0 & \textit{otherwise} \end{cases}$$

Where f_{ij} denotes the unary function labeled on the edge (v_i, v_j).

[2]Linear property is a special case of doubling closure property defined in the full paper. There we show that doubling closure property is more general for supporting Compress operation defined for parallel abstracted circuit contraction.

Computational circuit parallel evaluation can be done by repeated applications of the following three operations.

```
**Eval$_L$:**
For all nodes $v$ in the circuit whose children $w_1, \ldots, w_k$ are leaves, do
    $value(v) \leftarrow \odot\, (U_{w_1 v}(value(w_1)), \ldots, U_{w_k, v}(value(w_k)))$
    $U_{w_i v} \leftarrow 0$.

**Eval$_E$:**
for all pairs of nodes $v, u$ where $v$ is an expensive node with children $w_1, w_2$ in which one of them is a leaf and $u$ is a parent of $v$. ($w$ stands for the nonleaf child)
    find $h_{w,v,u} \in \mathcal{F}$ such that:
        $h_{w,v,u}(x) = U_{vu}(U_{w_1 v}(x) \otimes U_{w_2 v}(\underline{value}(w_2)))$
        if $w_2$ is leaf
        $h_{w,v,u}(x) = U_{vu}(U_{w_1 v}(\underline{value}(w_1)) \otimes U_{w_2 v}(x))$
        if $w_1$ is leaf.
    $U_{vu} \leftarrow 0$
for all pairs $w, u$.
    $g_{w,u} \leftarrow \oplus_v(h_{w,v,u})$; The sum is over defined $h_{w,v,u}$.
    $U_{wu} \leftarrow U_{wv} \oplus g_{w,u}$.
```

MM$_C$:

$$U \leftarrow U(X,C) \circ U(C,C) \oplus U(X,X)$$

Where $A \oplus B$ and $A \circ B$ are matrices addition and multiplication over ring $\{\mathcal{F}, \oplus, \circ\}$.

Claim 2.3 *The application of operations Eval$_L$, Eval$_E$ and MM$_C$ preserve the values of all nodes in the circuit.*

[PROOF] This claim straightforwardly follows the definitions of those three operations and the closure properties defined in previous subsection and can be proved by induction easily. □

```
**ALGORITHM** Parallel Circuit Eval
  **input** circuit $C$;
  **output** The value of the output nodes;
    **repeat**
      MM$_C$; Eval$_L$; Eval$_L$; Eval$_E$;
    **until** all nodes are leaf nodes;
```

Claim 2.4 *At the end of the algorithm, every node gets its value.*

Theorem 2.1 (Generic Theorem) *Let C be a circuit over domain D, operator set OP, unary function class $\mathcal{F}$. If there is a unary function class $\mathcal{F}'$ contains $\mathcal{F}$, $\mathcal{F}'$ is closed over the pair of operation sets (cheap operation, expensive operation), and the time, processors required to compute composition, combination, projection are bounded by $T(n)$ and $P(n)$ respectively, then C can be evaluated in $O(T(n) \cdot (log^2(n) + (\log n) \cdot (logd)))$ time, using $O(P(n) \cdot M(n))$ processors. Where n is the node size of C, d is the degree of C.*

This theorem demonstrates the relationships between parallel circuit evaluation and all the parameters of circuit. The constraints that all the computations of composition, combination, projection during the evaluation of the circuit must be bounded by small $T(n)$ and $P(n)$ show that the class unary functions need not only to be closed, but also to be uniform. In another word, we can find a reasonable data structure for the representing, manipulating of those functions efficiently in parallel. These will be discussed in later sections.

[PROOF] Note that if we replace all cheap nodes with addition nodes and all expensive nodes with multiplication nodes, then the circuit is converted into an arithmetic circuit of same size and degree. The theorem is proven by using the similarity between our algorithm and the algorithm for arithmetic circuit presented in [MiRK85] and the proof of Theorem 5.3 in [MiRK85]. □

3 Reduction Lemma

Lemma 3.1 (Reduction Lemma) *If $\mathcal{F} = \{L(x)\}$ is a class of monotonically increasing unary functions over a domain $D \subseteq \mathcal{R}$ and it is closed under composition and projection over OP_1, then $\mathcal{F}_{min} = \{min(L_1(x), ..., L_i(x)) \mid i \in \mathcal{N}, L_j(x) \in \mathcal{F}_1\}$ is closed over the pair of operations $(\{min\}, OP_1)$. $\mathcal{F}_{max} = \{max(L_1(x), ..., L_i(x)) \mid i \in \mathcal{N}, L_j(x) \in \mathcal{F}_1\}$ is closed under the pair of operations $(\{max\}, OP_1)$.*

[PROOF] Let $f(x) = min(L_1(x), ..., L_i(x))$, $g(x) = min(L'_1(x), ..., L'_j(x))$, $L_1(x), ..., L_i(x)), L'_1(x), ..., L'_j(x)) \in \mathcal{F}_1$.

- Since all functions in $\mathcal{F}_1$ are monotonicly, increasing, then:
  $$\begin{aligned} g \circ f(x) &= min(L'_1(f(x)), ..., L'_j(f(x))) \\ &= min(L'_1(L_1(x)), ..., L'_1(L_i(x)), ..., L'_j(L_i(x))) \end{aligned}$$
  Since $\mathcal{F}_1$ is closed composition, hence, $\mathcal{F}_{min}$ is closed under composition.
- $\mathcal{F}_{min}$ is closed under combination over $\{min\}$ since: $min(f, g) = min(L_1(x), ..., L_i(x), L'_1(x), ..., L'_j(x))$
- $\mathcal{F}_{min}$ is closed under distributing over $\{min\}$ since all functions in $\mathcal{F}_1$ are monotonicly increasing.
- $\mathcal{F}_{min}$ is closed under projection over OP_1, since $\mathcal{F}_1$ is closed under projection under OP_1 and $\mathcal{F}_{min}$ is closed under composition.

By duality, the second part of the lemma is proved. □

Using the Reduction Lemma, we have the following results.

- Since $\mathcal{F}_{(max,+)} = \{max(x,a) + b \mid a,b \in \mathcal{R}\}$ are closed under composition, projection over $\{max,+\}$, and all functions in it are monotonicly increasing, hence $\mathcal{F}_{(min,max,+)} = \{min(L_1(x),...,L_i(x)) \mid i \in \mathcal{N}, L_1,...,L_i \in \mathcal{F}_{(max,+)}\}$ is closed under the pair of operation sets $(\{min\},\{max,+\})$.
- $\mathcal{F}_{(min,max,\times)} = \{min(L_1(x),...,L_i(x)) \mid i \in \mathcal{N}, L_1,...,L_i \in \mathcal{F}_{(max,\times)}\}$ is closed under the pair of operation sets $(\{min\},\{max,\times\})$. Where $\mathcal{F}_{(max,\times)} = \{max(ax,b) \mid a,b \in \mathcal{R}^+\}$.
- $\mathcal{F}_{(min,max,+,\times)} = \{min(L_1(x),...,L_i) \mid i \in \mathcal{N}, L_1,...,L_i \in \mathcal{F}_{(max,+,\times)}\}$ is closed under the pair of operations $(\{min\},\{max,+,\times\})$. Where $\mathcal{F}_{(max,+,\times)} = \{max(a, b \cdot x) + c \mid a,b,c \in \mathcal{R}^+\}$.

4 Min-max-plus Circuits

We know of no previous results on the parallel evaluation of circuits with three or more operators. In this section, we will present some systematic methods for the design of parallel evaluation algorithms for thoses problems. We use as our example the min-max-plus circuit. (i.e. we consider computational circuits over domain $\mathcal{R}$, the set of real numbers, and operator set $\{min, max, +\}$.) Min-max-plus circuits (and/or min-max-plus trees) are important for computational AI, especial in game theory. Moreover, it seems that a lot of dynamic programming problems can be reduced to min-max-plus circuit problems). Any pair of operators is easy to evaluate in parallel since $\{\mathcal{R}, min, max\}$ forms an boolean algebra, and $\{\mathcal{R}, min, +\}$ and $\{\mathcal{R}, max, +\}$ form commutative rings. We consider the complexity of evaluating min-max-plus circuit in parallel.

4.1 Unary Functions

It is natural to view {min, max} as the cheap operators and {+} as the expensive operators. However, if $\mathcal{NC} \neq \mathcal{P}$, then there is no class of unary functions which is uniformly closed under the pair of operations $(\{min, max\},\{+\})$. By this observation, we take min (or max) as the cheap operator.

The class of unary functions $\mathcal{F}$ for min-max-plus is defined as follows:

Definition 4.1 (Unary function of min-max-plus) *Let $L_{a,b}$ denote the linear form $max(a + x, b) = a + x \wedge b$ and, for all $i \in \mathcal{N}$, let $L_{a_1,b_1} \vee \cdots \vee L_{a_i,b_i}$ denote $min(L_{a_1,b_1}(x),...,L_{a_i,b_i}(x))$. The class of unary functions of min-max-plus is defined as:*

$$F = \bigcup_{i=1}^{+\infty} \{L_{a_1,b_1} \vee \cdots \vee L_{a_i,b_i} \mid a_1,\ldots,a_i,b_1,\ldots,b_i \in R\}$$

We call this the **linear form representation** *of f. The* **size of the representation** *is the number of linear forms presented in the representation. The* **minimum representation** *of f is a linear form representations containing the minimum number of forms.*

We state as a claim the fact that F is the class of functions we will need for parallel evaluation of the min-max-plus circuits:

Claim 4.1 *$\mathcal{F}$ is closed over the pair of operator sets $(\{min\},\{max,+\})$.*

These unary functions allow us to partially evaluate parts of the circuit. They allow us to replace pieces of the circuit with a single edge that has an unary function on it. The circuits we obtain are defined below:

Definition 4.2 *A* **restricted CE-circuit** *in one free variable x is a CE-circuit C such that:*

1. *There exist an unique leaf node whose value is the variable x.*
2. *Every expensive node has at most two children one of which is a leaf.*

We say that C computes the unary function f at the node v if the value of the circuit C at the node v as a function of x is $f(x)$. Thus C computes a function for each node. If C has a distinguished output node we will simply say that f is the function computed by C, denoting f by f_C.

The main question we will consider in this and the next section is the size (to be defined in the next subsection) of f as a function of the size of original circuit. This size will effect the time and number of processors we need to use to evaluate the circuit.

Lemma 4.1 *Any unary function f produced during the evaluation of a CE-circuit C by repeated application procedure Parallel-Circuit-Eval is computable by a restricted CE-circuit obtain from C by deletion, partial evaluation, and shunting. Thus f is computable by a restricted circuit of size at most $|C|$.*

[Proof] The proof is by induction on the number a applications of Parallel-Circuit-Eval. For no application the procedure the Lemma is clearly true. When we apply Eval$_L$ or Eval$_E$ we are simply either deleting part of the circuit or doing a partial evaluation. When we apply Compress$_C$ we are combining two subcircuits under composition to get each unary function, shunting. □

4.2 Uniformity of $\mathcal{F}$

Definition 4.3 (Size of functions in $\mathcal{F}$) *The size function $f \in \mathcal{F}$ is the number of forms in its minimum linear form representation, denoted by $\underline{size}(f)$.*

During the evaluation of the circuit we shall need to take an arbitrary representation and reduce it to a minimum representation.

Lemma 4.2 *A linear form representation of size n can be reduced to minimum one in $O(\log n)$ time using n processors.*

[PROOF] Let $L_{a_1,b_1} \vee \cdots \vee L_{a_n,b_n}$ be an arbitrary linear form representation. We first sort the forms on their a values. Since sorting can be performed in $O(\log n)$ time using n processors [AjKS83,ReVa87] we may assume that the the a_i are already sorted and distinct. For the ith linear form see assign it the value $v_i = b_i - a_i$. It follows that a linear form L_{a_i,b_i} is in the minimum representation if v_i is strictly greater than v_j for $1 \leq j < i$. Thus using all prefix sums we can compute the values v_i that will be in the minimum representation, and therefore the forms in the minimum representation. □

Given a sorted minimum representation for a function f is follows that we can compute the value of f at x in constant time using n processors or in $O(\log n)$ time using one processor. We show that the minimum representation of the unary function is not too large. The **breakpoints** of a linear form function f is the set of a's in the minimum representation of f.

Definition 4.4 *The* **breakpoints** *of a restricted CE-circuit C for the operations* $(\{\min\}, \{\max, +\})$ *is the union over breakpoints of all functions computed by C.*

Lemma 4.3 *The number of breakpoints of a restricted CE-circuit over* $\{\min\}, \{\max, +\})$ *is bounded by the sum of number of max nodes and the total number of breakpoints of the edge functions of C.*

[PROOF] We shall only prove the case when the functions on the edge are all the identity. The general case is not much harder. The proof is by induction on the size of the restricted CE-circuit. A single node is a restricted CE-circuit and the only function it computes is a constant. Thus it has no breakpoints which is as claimed in the Lemma.

Suppose the Lemma is true for all circuits of size n or less. We must prove the Lemma for circuits of size $n+1$. Let C be such a circuit of size $n+1$ and v one of its output nodes. If we remove v from C and all the edges associated with it we have a circuit C' of size n, possibly a disconnected circuit. Thus by induction the Lemma holds for C'. To prove the Lemma for C we consider three case depending on whether v is a min ,max, or plus node. Let $v_1, \ldots, v_k$ be the children of v.

First suppose that v is a min node. In this case the value computed at v is just the min of the values computed at $v_1, \ldots, v_k$. Thus the breakpoints of v is at most the union of the breakpoints of $v_1, \ldots, v_k$. Thus, no new breakpoint have been introduced by v.

Second, suppose that v is a max node. In this case v has at most two children v_1, v_2 where v_1 is a leaf with value a and v_2 computes the value $f(x)$. It follows that the max of the constant function a with f will introduce at most one new breakpoint, the point where the curve a intersects the monotone increasing curve f. If they intersect in a line then no breakpoints are introduced. Since C has one more max node than C' this case also follows.

Third, suppose that v is a plus node. Here, the value of v is the sum of a constant a and a function f. But the breakpoints of f are unchanged by translation of f by an additive constant. Thus, the breakpoints of the function computed by v are just the breakpoints of f. Therefore the breakpoint of C are the same as the breakpoints of C'. □

Lemma 4.4 *If C be a* **min-max-plus** *circuit with m max nodes and e edges and all unary functions on the edges of C have size less than constant c then the size of an unary function used by Parallel-Circuit-Eval during the evaluation of C will have size at most* $c \cdot e + m$. *In the case when the edge functions are trivial the size of the function are all bounded by m.*

[PROOF] By Lemma 4.1 every unary function is computed by a subcircuit of C. Further by Lemma 4.3 the number of breakpoints in any restricted CE-subcircuit is bounded by the number of breakpoints contributed by the edge function, which is at most $c \cdot e$, plus the number of max nodes, m. □

In order to finish the description of the algorithm and the proof that it works efficiently we must show how to perform matrix multiplication of matrices whose entries are unary functions given in minimum linear form representation. The product is over the semiring $\{\min, \textit{composition}\}$. We assume the matrices are of size n and total number of breakpoint of all the function including those in the product is bounded by n. Thus we must show how to (1) perform composition in $O(\log n)$ time and n processors (2) computed the min of n functions in the same time and processor count.

Lemma 4.5 *If f and g are two functions of size at most n then their composition and minimum representation can be computed in $O(\log n)$ time and n processors. In general the composition of any two monotone piecewise linear functions each with n breakpoints can be computed in O(logn) and n processors.*

[PROOF] See paper [MiTe87]. □

Lemma 4.6 *The minimum of a set of n functions each of size at most n can be computed in O(logn) time using n^2 processors.*

[PROOF] This Lemma follows by simply concatenating the representations of the n functions together and then appling Lemma 4.2 we get a minimum representation for the answer. □

Theorem 4.1 *Any* **Min-max-plus** *circuit C with m max nodes, n nodes, and degree d can be evaluated in $O((\log n)(logdn)))$ time using $O(n \cdot M(n))$ processors.*

Remark 6.1. In section 5, we show that $\mathcal{F}_{(min,max,\times)}$ is closed over $\{min, max, \times\}$. Using the similar uniformity proof in this section, we can prove that $\mathcal{F}_{(min,max,\times)}$ is uniformly closed. Therefore, we have the following theorem.

Theorem 4.2 *Any* **Min-max-times** *circuit C over $\mathcal{R}^+$ with m max nodes, n nodes, and degree d, can be evaluated in $O((\log n)(logdn)))$ time using $O(n \cdot M(n))$ processors.*

Remark 6.2. In full paper, we show that there is a class of unary functions $\mathcal{F}_1$ which is closed over the pair of operations $(\{min\}, \{max, plus\})$ as well as the pair of operations $(\{max\}, \{min, plus\})$. Moreover, it is uniform closed. Therefore, we can have algorithm similar to the parallel Boolean circuit evaluation algorithm which has dynamic adaptivity for evaluate min-max-plus circuits.

5 Boolean Circuits and Their Parallel Evaluation

We will present an uniform algorithm to evaluate Boolean circuits with unary functions $\mathcal{F} = \{(a \wedge x) \vee b \mid a, b \in \mathcal{D}\}$ without any preprocessing using the methods developed in last section. The time complexity T_C of this algorithm always satisfies $T_C \leq O((\log n)(\log n(min(d_{\wedge\vee}, d_{\vee\wedge})))$. Moreover, we show that there are Boolean circuits with exponential $d_{\wedge\vee}$ and $d_{\vee\wedge}$ which can be evaluated in polylogarithmic time using $M(n)$ processors processors without knowing the structure of the circuit beforehand.

Definition 5.1 (Boolean Circuit) *A Boolean circuit over Boolean algebra $\mathcal{B} = \{\mathcal{D}, \vee, \wedge\}$ is a computational circuit $\{\mathcal{D}, \mathcal{OP}, \mathcal{F}, \mathcal{C}\}$, where $\mathcal{OP} = \{\vee, \wedge\}$, $\mathcal{F} = \{(a \wedge x) \vee b \mid a, b \in \mathcal{D}\}$.*

Boolean Algebra $\mathcal{B} = \{\mathcal{D}, \vee, \wedge\}$ itself forms a commutative semi-ring and therefore as a consequence of results in [MiRK85], the simple Boolean circuit can be evaluated in $O((\log n)(\log nd))$ times with $M(n)$ processors. But this is not a tight bound for the following reason.

- It is not clear about the definition of degree of Boolean circuit since we can view $\vee$ or $\wedge$ as addition in $\mathcal{B}$. The corresponding degree is denoted by $d_{\wedge\vee}$ and $d_{\vee\wedge}$ respectively. One straightforward way to apply the algorithm in [MiRK85] to Boolean circuit is to compute $d_{\wedge\vee}$ and $d_{\vee\wedge}$ first and choose the operator with bigger degree as addition in ring. Another way is to make two copys of the circuit and evaluate one circuit using $\vee$ as addition and another using $\wedge$ as addition. However, both of those methods are not uniform in the sense that in the first one we have to introduce computation over $+$ and it is not easier to compute the degree that evaluate the circuit. In second we have to coordinate two evaluation processes. Moreover, if $d_{\wedge\vee}$ and $d_{\vee\wedge}$ are both exponential, we can not deduce a polylogarithmic algorithm. Moreover, in order to use previous results, $\vee$-node or $\wedge$-node must has in-degree two.

5.1 Closure Properties

Lemma 5.1 (Duality Lemma) *For all $a, b \in \mathcal{D}$ in a Boolean algebra $\{\mathcal{D}, \vee, \wedge\}$, there is $c \in \mathcal{D}$, such that:*

$$(a \wedge x) \vee b = (b \vee x) \wedge c$$

Lemma 5.2 (Boolean circuit lemma) *$\mathcal{F} = \{(a \wedge x) \vee b \mid a, b \in \mathcal{D}\}$ is closed over the pair of operation sets $(\{\vee\}, \{\wedge\})$ as well as the pair of operation sets $(\{\wedge\}, \{\vee\})$.*

[PROOF] Is is suffice to prove that $\mathcal{F}$ is closed over the pair of operations sets $(\{\vee\}, \{\wedge\})$ due to duality lemma.
[PROOF] For all $f_{(a,b)}(x) = (a \wedge x) \vee b$
(1) $\mathcal{F}$ is closed under composition since $f_{(a_2,b_2)}(f_{(a_1,b_1)}(x)) = f_{(a_1 \wedge a_2, a_2 \wedge b_1 \vee b_2)} \in \mathcal{F}$
(2) $\mathcal{F}$ is closed under combination over $\{+\}$ since $f_{(a_1,b_1)} \vee f_{(a_2,b_2)} = f_{(a_1 \vee a_2, b_1 \vee b_2)} \in \mathcal{F}$
(3) $\mathcal{F}$ is linear over $\{\vee\}$, since $f_{(a,b)}(x \vee y) = f_{(a,b)}(x) \vee f_{(a,b)}(y)$.
(4) $\mathcal{F}$ is closed under projection over $\{\wedge\}$ since for all $c \in \mathcal{D}$, $f_{(a,b)}(c \wedge x) = f_{(a \wedge c, b)}(x)$ and '$\wedge$" is commutative. □

5.2 The Algorithm

Because of duality lemma, there is no reason to view one of $\{\vee, \wedge\}$ as cheap operator and another as expensive operator. And since the general Boolean circuit evaluation problem is P-Complete [Ladn75], one of them must be expensive. However, Boolean circuit lemma tells us that we can do Eval$_E$ and MM$_C$ on $\vee$ nodes as well as on $\wedge$ node (we denote by Eval$_\vee$, Eval$_\wedge$, MM$_\vee$, MM$_\wedge$ respectively). This provides an outline for Boolean circuit parallel evaluation: Define two parallel evaluation phases, one takes $\vee$ as expensive operation, another takes $\wedge$ as expensive operation. We denote those two phases by $Phase(\wedge, \vee)$ and $Phase(\vee, \wedge)$ respectively. The algorithm is defined as the repeated applications of $Phase(\wedge, \vee)$ and $Phase(\vee, \wedge)$ alternatively. In order to make this work, we have to solve the following three problems: (1) Unbounded fan-in of expensive nodes. (2) The problem caused by edges from expensive node to expensive node. (3) The problem caused by alternation.

We now define an operation to deal with unbounded fan-in of expensive nodes.

Trimming:
For all nodes v in the circuit whose children $w_1, \ldots, w_k$ are all leaves, **do**

```
$value(v) \leftarrow \odot\,(U_{w_1v}(value(w_1)), \ldots, U_{w_kv}(value(w_k)))$
$U_{w_iv} \leftarrow 0$. Where $\odot$ is the operation of $v$.
For all nodes $v$ with label $\odot$ and children $w_1, \ldots, w_k$,
w.l.o.g let $w_1, \ldots, w_t$ be leaf nodes, where $t < k$, do
    $a \leftarrow \odot(U_{w_1v}(value(w_1)), ..., U_{w_tv}(value(w_t)))$
$U_{w_{t+1}u}(x) \leftarrow a \odot U_{w_{t+1}u}(x)$; $U_{w_1v}, ..., U_{w_tv} \leftarrow 0$
```

Graphically, the **Trimming** operation simply disconnect all leaves from their parents. This operation can be done in $O(\log n)$ time, using n^2 processors. It is easy to show that after the application of Trimming, the value of each node is not changed.

Trimming will in general introduce many nodes with fan-in 1 and these nodes of fan-in 1 form a forest as a subgraph in the circuit. (see Figure 1).

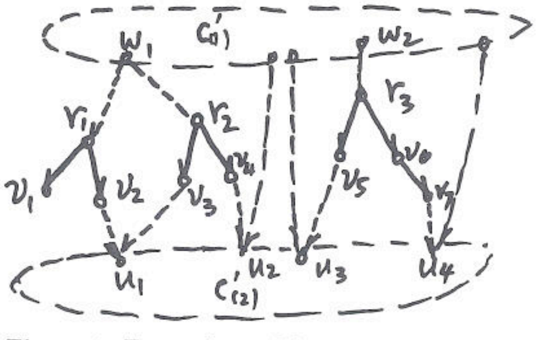

Figure 1: Forest formed by nodes of fan-in one

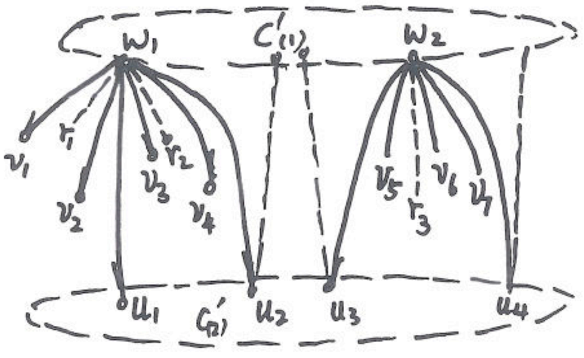

Figure 2: Forest Removal for the example in Figure 1

We will remove the forest and replace it by equivalent unary functions. The equivalent unary functions of all nodes in the forest are the composition of the unary functions from the child of its root to itself (see Figure 2). By using the parallel tree contraction technique developed in [MiRe85], we can find the equivalent unary functions for all nodes in the forest in $O(\log n)$ time, using only n processors. We name this function by **Forest-Removal** which defined as following:

```
Forest-Removal:
for all $v$ in the forest, Find the equivalent unary function
$f_v$ in the forest using tree contraction technique. Let $r$
denote the root of $v$ and $c(r), c(v)$ be the childs of $r$ and $v$
in the circuit respectively.
$U_{c(v)v} \leftarrow 0$; $U_{c(r)v} \leftarrow f_v$;
for all pairs of nodes $w$, $u$ such that $w$ is child of some
nodes which are root in the forest and $u$ is of operation
$\odot$ and some of whose children $v_1, \ldots, v_k$ are nodes in the
forest rooted by some parent of $w$ do
$g_u \leftarrow \odot_{i=1}^{k}(U_{v_iu} \circ f_{v_i})$; $U_{c(r)u} \leftarrow U_{c(r)u} \odot g_u$; $U_{v_iu} \leftarrow 0$.
```

Claim 5.1 *After the application of* **Forest-Removal**, *there is no node which is not an output node and which has fan-in 1. And the value of each node is preserved by* **Forest-Removal** *operation. Moreover,* **Forest-Removal** *takes $O(\log n)$ time, using n^2 processors.*

Now, we are ready to specify the two important phases and the algorithm for boolean circuit parallel evaluation.

```
Phase(∨, ∧)
  Trimming; Trimming; Trimming;
  Forest-Removal; MM∨;

Phase(∧, ∨)
  Trimming; Trimming; Trimming;
  Forest-Removal; MM∧;

ALGORITHM Parallel Boolean Circuit Eval
  Forest-Removal;
  repeat
    Phase(∨, ∧); Phase(∧, ∨);
  until all nodes are output nodes;
```

Lemma 5.3 *After the application of Phase($\vee, \wedge$) (or Phase($\wedge, \vee$)), All the non-output nodes have fan-in at least two.*

[**PROOF**] Clearly, after the application of Forest-Removal, all non-output nodes have fan-in at least two. For all non-output nodes v, MM$_\vee$ either makes no change on v or connect all the children of some of its children to it. Hence, in any case, after the application of MM$_\vee$, v has at least two children. □

Definition 5.2 *The $\wedge\vee$-height of a node v denoted by $\underline{h}_{\wedge\vee}(v)$ in circuit is defined inductively:*

- *If v is a leaf, then $\underline{h}_{\wedge\vee}(v) = 1$.*
- *If v is an $\vee$-node then: $\underline{h}_{\wedge\vee}(v) = \sum_{u \in C(v)} \underline{h}_{\wedge\vee}(u)$.*
- *If v is a $\wedge$-node, then: $\underline{h}_{\wedge\vee}(v) = max(max_{u \in CC(v)} \underline{h}_{\wedge\vee}(u) + 1/2, max_{u \in EC(v)}(\underline{h}_{\wedge\vee}(u)))$ where $C(v)$, $CC(v)$ and $EC(v)$ are the set of children, $\wedge$-children and $\vee$-children or leaf children of v. respectively.*

The $\wedge\vee$-height of a circuit is the maximum height over its nodes. A child w of a $\wedge$-node v is dominant if either: w is an vee-node and $\underline{h}_{\wedge\vee}(w)=\underline{h}_{\wedge\vee}(v)$ or w is a $\wedge$-node and $\underline{h}_{\wedge\vee}(w)+1/2=\underline{h}_{\wedge\vee}(v)$. The $\vee\wedge$-height (denoted by $\underline{h}_{\vee\wedge}(v)$) is defined in the similar way.

Claim 5.2 *The $\wedge\vee$-height (or $\vee\wedge$-height) of a circuit is less than $e \cdot d_{\wedge\vee} + d_{\wedge\vee}$ ($e \cdot d_{\vee\wedge} + d_{\vee\wedge}$).*

[PROOF] Proven in Theorem 4.2 from [MiRK85]. □

From now on, let C denotes a Boolean circuit in which all non-output nodes in it has fan-in at least two.

Claim 5.3 *Let C' be the circuit after the application of Phase($\vee, \wedge$) on C and v' be the image of v. If v' is a leaf and there exist at least one $\wedge$-node u' such that (v', u') is an edge in C', then v has $\vee\wedge$-height at least two.*

[PROOF] Suppose the $\vee\wedge$-height of v is less than two, there are two cases: If its height is one, then v is either a leaf or all its children are leaves. Hence, after the first Trimming operation, v becomes a leaf. If its height of v is $3/2$, then v is must be a $\vee$ node whose children are either leaves or $\vee$-nodes with all leaf children. So the second Trimming makes v a leaf. Therefore, in each case, the application of the third Trimming disconnects v' and u'. Contradiction. □

Lemma 5.4 *Let C' be the circuit after the application of Phase($\vee, \wedge$) on C and v' be the image of v. Then $\underline{h}_{\vee\wedge}(v') \leq \underline{h}_{\vee\wedge}(v)/2$.*

[PROOF] The proof is done by induction on the length of the longest path from a leaf : the level of the node.
Suppose that all the children of v' are leaves:

- If v' is a $\wedge$-node, then by definition, the height $\underline{h}_{\vee\wedge}(v')=k$, where k is the in-degree of node v'. By Claim 5.3, $\underline{h}_{\vee\wedge}(v) \geq 2k$,
- If v is a $\vee$-node then by definition, $\underline{h}_{\vee\wedge}(v')=1$. Suppose $\underline{h}_{\vee\wedge}(v) \leq 3/2$, then after the second Trimming operation, v become a leaf, contradiction.

Suppose that all the children of v' are either leaf nodes or internal nodes in C' satisfy the lemma:

- If v is a $\wedge$-node then the lemma follows by Claim 5.3 and the assumption.
- If v is a $\vee$-node, then let u' be a dominant child of v'. If u is a $\wedge$-node, then it is straightforward. Now suppose u is a $\vee$-node. Since u and v are all $\vee$-nodes, there must be other nodes in between one of the paths from u to v, otherwise, u' will not be a child of v' after the application of the $\mathrm{MM}_{\vee}$ operation. Moreover, all nodes has fan-in at least two. Thus by the definition of height, $\underline{h}_{\vee\wedge}(u)+1 \leq \underline{h}_{\vee\wedge}(v)$. Hence: $\underline{h}_{\vee\wedge}(v) \geq 2 \cdot \underline{h}_{\vee\wedge}(v')$. □

Lemma 5.5 *For all nodes v in Boolean circuit C, the application of Phase($\vee, \wedge$) (Phase($\wedge, \vee$)) will not increase $\underline{h}_{\wedge\vee}(v)$ ($\underline{h}_{\vee\wedge}(v)$).*

[PROOF] It is proven by simple induction on the level of nodes. See the full paper. □

Theorem 5.1 *A Boolean circuit over a Boolean algebra $\mathcal{B} = \{\mathcal{D}, \vee, \wedge\}$ can be evaluated in time no more than $O((\log n)(\log n(min(d_{\wedge\vee}, d_{\vee\wedge})))$ using $M(n)$ processors. Moreover there are Boolean circuits with exponential $d_{\wedge\vee}$ and $d_{\vee\wedge}$ which can be evaluated in polylogarithmic time using $M(n)$ processors without knowing the structure of the circuit beforehand.*

[PROOF] The first part of the theorem is a consequence of lemma 5.4, 5.5. The second part follows the fact that for all node v in a Boolean circuits C,if at least one of $d_{\wedge\vee}(v)$ and $d_{\vee\wedge}(v)$ is no more than $O(2^{(\log n)^*})$, then v will get its value in polylogarithmic time. And it is easy to construct Boolean circuits of exponential $d_{\wedge\vee}$ and $d_{\vee\wedge}$, but all nodes v in it have one of $d_{\wedge\vee}(v)$ and $d_{\vee\wedge}(v)$ is no more than $O(2^{(\log n)^*})$. □

The beauty of this parallel Boolean circuit evaluation lies in its dynamic adaptability to the structure of the circuit. This power comes from the symmetrical structure of the unary function class. However, It is still unknown about the tight bound on the running time of this parallel Boolean circuit evaluation algorithm and some stronger characterizations of the circuits which can be evaluated in polylogarithmic time. However, we can use this algorithm define the dynamic degree of a Boolean circuit.

Definition 5.3 (Dynamic Degree) *If Boolean circuit C can be evaluated by using the above algorithm in T_C times, then dynamic degree of C (denoted by d_C) is defined as $d_C = 2^{T_C/(\log n) - \log n}$.*

6 Parallel Algorithms for Circuits over Noncommutative Semi-Ring

In this section, we develop a theory for parallel noncommutative semi-ring circuits evaluation via the uniform closure properties of unary function classes and show all circuits over finite noncommutative semi-ring and circuits over infinite noncommutative semi-ring which has finite dimension over a commutative semi-ring can be evaluated in polylogarithmic time in its size and degree using $M(n)$ processors. This provides a partial answer to the open problem in [MiRK85]. Moreover, this demonstrates evidence that the parallel circuit evaluation using uniform closure properties is more general than when simply using the commutative ring condition even in the case of two operators.

6.1 General Case

Let $\mathcal{NR} = \{\mathcal{D}, \oplus, \odot\}$ be a noncommutative semi-ring.

Definition 6.1 (Unary Function Class) *Let:*

$$\mathcal{F} = \{\Sigma_{i=1}^{k} a_i \odot x \odot b_i \mid k \in \mathcal{N}, a_i, b_i \in \mathcal{D}\}$$

We call each $a_i \odot x \odot b_i$ be item. We define the size of a function in $\mathcal{F}$ be the minimum number of items define it.

Theorem 6.1 *$\mathcal{F}$ is closed over the pair of operation sets $(\{\oplus\},\{\odot\})$.*

[PROOF] The proof is straightforward and given in full paper. □

Let $\mathcal{INR} = \{\mathcal{D},\oplus,\odot\}$ noncommutative semi-ring with infinite domain $\mathcal{D}$, we say $\mathcal{INR}$ is **noncombinable** if for all $a,b,c,d \in \mathcal{D}$, there exist $s,t \in \mathcal{D}$ such that $(a \odot x \odot b) \oplus (c \odot x \odot d) = s \odot x \odot t$ iff $a = b$ or $c = d$. In the full paper, we will give a constructive proof that for all noncombinable infinite noncommutative semi-ring. Let $\mathcal{INR} = \{\mathcal{D},\oplus,\odot\}$, $\mathcal{F}$ is not uniformly closed over the pair of operation sets $(\{\oplus\},\{\odot\})$, i.e. there exist a restricted CE-circuit C over $\mathcal{INR}$, generate a function in $\mathcal{F}$ whose size is exponential in the size of the circuit. However, we will show there are infinite noncommutative semi-rings, such that all circuits over it can be evaluated in logarithmic time in its degree and size in next subsection.

6.2 Finite Case

Theorem 6.2 *For all finite noncommutative semi-ring $\mathcal{FNR} = \{\mathcal{D},\oplus,\odot\}$, for all computational circuit C over $\mathcal{FNR}$ with size n and degree d, C can be evaluated in $O((\log n)(\log dn))$ time using $M(n)$ processors on CRCW PRAMs.*

[PROOF] Let the unary function class for C be:

$$\mathcal{F} = \{\Sigma_{i=1}^{k} a_i \odot x \odot b_i \mid k \leq \mid \mathcal{D} \mid^2, a_i, b_i \in \mathcal{D}\}$$

It is suffice to prove $\mathcal{F}$ is closed over the pair of operation sets $(\{\oplus\},\{\odot\})$. Clearly, $\mathcal{F}$ is linear over $\oplus$ and $\mathcal{F}$ is closed under projection over $\odot$. Let $f(x) = \Sigma_{i=1}^{k_1} a_i \odot x \odot b_i$ and $g(x) = \Sigma_{i=1}^{k_2} c_i \odot x \odot d_i$. $f \oplus g = (\Sigma_{i=1}^{k_1} a_i \odot x \odot b_i) \oplus (\Sigma_{i=1}^{k_2} c_i \odot x \odot d_i)$. If $k_1 + k_2 \leq \mid \mathcal{D} \mid^2$, then clearly $f \oplus g \in \mathcal{F}$. Otherwise, since $\mathcal{D}$ is finite and there are at most $\mid \mathcal{D} \mid^2$ different items, there must exist a $t \leq \mid \mathcal{D} \mid^2$, $u_1,\ldots,u_t,v_1,\ldots,v_t \in \mathcal{D}$ such that $f \oplus g = \Sigma_{i=1}^{k_1} u_i \odot x \odot v_i$. So, $\mathcal{F}$ is closed under combination. Similarly, we can prove that $\mathcal{F}$ is closed under composition. □

6.3 Examples of Finite Noncommutative Ring

Let $G = \{V,\Sigma,P,S\}$ be a context-free grammar in Chomsky normal form. We define the domain $\mathcal{D}_G = 2^{(V-\Sigma)}$, and operator set $\mathcal{OP}_G = \{\cup,\lhd\}$, where $\cup$ is the conventional union operator in set theory and $\lhd$ is a binary operator such that For all $S,T \in \mathcal{D}$, $\lhd(S,T) = \{C \mid \exists A \in S, B \in T, C \leftarrow A,B\}$. The circuits we consider are defined over domain $\mathcal{D}_G$ and operator set $\mathcal{OP}_G$, we call them $CFL - Circuit$.

It is clearly that $\{\mathcal{D}_G,\cup\}$ forms group. $\{\mathcal{D}_G,\lhd\}$ forms a semi-group. Moreover, $\lhd$ is distributive over $\cup$. So, $\{\mathcal{D}_G,\cup,\lhd\}$ forms a semi-ring. Moreover, it is finite. But $\lhd$ is not commutative.

Corollary 6.1 (CFL-Circuit) *$CFL - Circuit$ can be evaluated in $O((\log n)\cdot(\log(n \cdot d)))$ time using $M(n)$ processors. Where n is the node size of the $CFL - Circuit$.*

Since CFLs recognition problem can be reduced to a $CFL - Circuit$ evaluation problem in polylogarithmic time and the corresponding $CFL - Circuit$ has polynomial size and degree to the length of input string in full paper. Therefore:

Corollary 6.2 *CFLs' recognition problem lies in NC^2.*

Note that Ruzzo [Ruzz81], [Ruzz80] first showed that there are Boolean circuits that simultaneously have polynomial size and polylogarithmic depth which recognize context free languages. Valiant et al [VSBR82] showed that the Boolean circuits defined by Cocke-Kasami-Younger algorithm has linear degree and therefore CFLs recognition lies in $\mathcal{NC}$. The difference between our solution and their solutions is that they break the operation of CFLs recognition down to Boolean operations, while we implement CFLs recognition using higher level, structured operations. We shall show the advantage of this approach in later paper.

We can also reduce the logical query programs [UlGe85] to a circuit whose size is a polynomial of the size of the program. And it is easy to prove if the basic logic program has polynomial fringe, then the degree of the circuit is a polynomial of the size of the program. So, we have:

Corollary 6.3 *A basic logic program with the polynomial fringe property is in NC^2.*

6.4 Semi-Ring Over Matrices

Let $\mathcal{ICR}$ be any commutative semi-ring and $\mathcal{NR} = \{\mathcal{D},\oplus,\odot\}$ be a noncommutative semi-ring. Where:

$$\mathcal{D} = \{A^{k\times k} \mid A_{ij} \in \mathcal{ICR}, 1 \leq i,j \leq k\}$$

Clearly, $\mathcal{NR}$ is a noncommutative semi-ring with infinite domain is $\mathcal{ICR}$ is infinite.

Theorem 6.3 *For all circuits C over $\mathcal{NR}$ with n nodes and degree d, C can be evaluated in $O((\log n)(\log nd))$ time, using $M(n)$ processors.*

[PROOF] (sketch) For all $A^{k\times k} \in \mathcal{D}$, $A^{k\times k}X^{k\times k}$ and $X^{k\times k}A^{k\times k}$ are linear transformation of $(x_{1,1},\ldots,x_{k,k})$ in space $\mathcal{ICR}^{k^2}$. Let $\mathcal{D}_1 = \{C^{k^2\times k^2} \mid C_{i,j} \in \mathcal{ICR}, 1 \leq i,j \leq k^2\}$. Therefore, There exist $B^{k^2\times k^2}, C^{k^2\times k^2} \in \mathcal{D}_1$ such that:

$$B^{k^2\times k^2} \oslash \begin{bmatrix} x_{1,1} & x_{1,2} & \cdot & x_{1,k} \\ x_{2,1} & \cdot & \cdot & x_{2,k} \\ \vdots & \vdots & \vdots & \vdots \\ x_{k,1} & x_{k,2} & \cdot & x_{k,k} \end{bmatrix} = B^{k^2\times k^2} \begin{pmatrix} x_{1,1} \\ x_{1,2} \\ \vdots \\ x_{k,k} \end{pmatrix}$$

$$= A^{k\times k} \begin{pmatrix} x_{1,1} & x_{1,2} & \cdot & x_{1,k} \\ x_{2,1} & \cdot & \cdot & x_{2,k} \\ \vdots & \vdots & \vdots & \vdots \\ x_{k,1} & x_{k,2} & \cdot & x_{k,k} \end{pmatrix}$$

$$C^{k^2\times k^2} \oslash \begin{bmatrix} x_{1,1} & x_{1,2} & \cdot & x_{1,k} \\ x_{2,1} & \cdot & \cdot & x_{2,k} \\ \vdots & \vdots & \vdots & \vdots \\ x_{k,1} & x_{k,2} & \cdot & x_{k,k} \end{bmatrix} = C^{k^2\times k^2} \begin{pmatrix} x_{1,1} \\ x_{1,2} \\ \vdots \\ x_{k,k} \end{pmatrix}$$

$$= \begin{pmatrix} x_{1,1} & x_{1,2} & \cdot & x_{1,k} \\ x_{2,1} & \cdot & \cdot & x_{2,k} \\ \vdots & \vdots & \vdots & \vdots \\ x_{k,1} & x_{k,2} & \cdot & x_{k,k} \end{pmatrix} A^{k\times k}$$

Hence, we define the following unary function class:

$$\mathcal{F} = \{C^{k^2\times k^2} \oslash X \mid C^{k^2\times k^2} \in \mathcal{D}_1\}$$

Lemma 6.1 *$\mathcal{F}$ is closed over the pair of operation sets $(\{\oplus\}, \{\odot\})$.*

[PROOF] In the full paper. □

7 Open Problems

- It is interesting and important to have a general theory for unary function class construction when a operator set is given and have theory for proving closure properties of certain classes of unary functions.
- The dynamic complexity of min-max-plus-times circuit over $\mathcal{R}^+$ is unknown even though we construct a class of unary functions which is closed over $(\{min\}, \{max, +, \times\})$ by means of reduction lemma in section 6. E.g the uniformity of $\mathcal{F}_{(min,max,+,\times)}$ is still open.
- Is there a operator set whose cheap operation has more than one operators?
- What is the tight bound on the running time of the parallel Boolean circuit evaluation algorithm?
- What is the dynamic complexity of circuit over a ring with division?

References

[AjKS83] M.Ajtai, J.Komlos, E.SZEMEREDI. Sorting in *clogn* Parallel Steps. *Combinatorica* 3(1) pp1-19, 1983.

[Ladn75] R.E.Ladner. The Circuit Value Problem is log Space Complete for P. *SIGACT News*, 7,1, pp18-20, 1975

[MiRe85] G.L.Miller, J.H. Reif. *Parallel Tree Contraction and its Application.* In FOCS 85, 1985.

[MiRK85] G.L.Miller, V.Ramachandran, E.Kaltofen. *Efficient Parallel Evaluation of Straight-line Code and Arithmetic Circuits.* Tech. Rept., Univ. of Southern Calif. 1985

[MiTe87] G.L.Miller, S-H.Teng. *Systematic Methods for Tree Based Parallel Algorithm Development.* In The Second International Conference on Supercomputing, 1987.

[ReVa87] J.H.Reif, L.G.Valiant. Logarithmic Timing Sort for Linear Size Network. *JACM* pp66-76, Jan.,1987.

[Ruzz80] W.L.Ruzzo. Tree-Size Bounded Alternation. *JCSS* pp218-235, Oct, 1980

[Ruzz81] W.L.Ruzzo. On Uniform Circuit Complexity. *JCSS* pp365-383, June, 1980

[UlGe85] J.D.Ullman, A.V.Gelder. *Parallel Complexity of Logical Query Programs.* TR, Stanford University 1985

[VSBR83] L.G.Valiant S.Skyum S.Berkowitz C.Rackoff. Fast Parallel Computation of Polynomials Using Few Processors. In *SIAM J.Computing*, pp641-644, vol12, No4, Nov. 1983