# Axiomatizing proof tree concepts in Bounded Arithmetic

**Satoru Kuroda**

**Gunma Prefectural Women's University**
**1395-1, Kaminote, Tamamura, Gunma, 370-1193, Japan**
satoru@gpwu.ac.jp

**Abstract**

We construct theories of Cook-Nguyen style two-sort bounded arithmetic whose provably total functions are exactly those in LOGCFL and LOGDCFL. Axiomatizations of both theories are based on the proof tree size characterizations of these classes. We also show that our theory for LOGCFL proves a certain formulation of the pumping lemma for context-free languages.

**1998 ACM Subject Classification** F.4.1 Mathematical Logic

**Keywords and phrases** Bounded Arithmetic, LOGCFL, LOGDCFL, Proof tree

**Digital Object Identifier** 10.4230/LIPIcs.CSL.2012.440

## 1 Introduction

The complexity classes LOGCFL and LOGDCFL are subclasses of P which gain a large popularity in the complexity theory community during the last two decades. The class LOGCFL consists of predicates logspace reducible to context free languages and LOGDCFL is defined in the same manner as LOGCFL but with deterministic context free languages.

LOGCFL lies between NL and $AC^1$ and LOGDCFL contains L but we do not know whether any of these inclusions is proper or not. Furthermore, we do not know the relationship between NL and LOGDCFL.

One of the main feature of LOGCFL is that it has several natural alternative characterizations such as NAuxPDA [9], alternating Turing machines [8] and semi-unbounded fan-in circuits [10]. Also LOGCFL has natural complete problems such as word problems for finite groupoids and acyclic conjunctive queries [4]. Moreover, it is worth noting that LOGCFL is closed under many operations including the closure under complementation [2].

On the contrary, much less is known about the class LOGDCFL. In particular, no natural complete problem is found so far.

In this paper we construct theories of bounded arithmetic for LOGCFL and LOGDCFL using their proof tree size characterizations. To author's knowledge, no such theory for LOGDCFL has been proposed so far. For LOGCFL, the author [6] defined a theory $V\text{-}Q^{SAC}$ which axiomatize the concept of $SAC^1$ circuits. However, the theory has an augmented language which represents generalized quantifier expressing that a $SAC^1$ circuit has an accepting tree on an input.

Our approach is very similar to the one developed by Cook and Nguyen [3], namely augmenting the theory $V^0$ by axioms expressing the concept of a given complexity class.

The concepts we use for constructing theories for LOGCFL and LOGDCFL are somewhat similar. That is, both concepts are based on circuits having polynomial proof tree size.

For LOGDCFL, McKenzie et.al. [7] proved that the class is identical to the class of predicates decidable by polynomial size Multiplex-Select circuits with polynomial proof tree

Computer Science Logic 2012 (CSL'12).
Editors: Patrick Cégielski, Arnaud Durand; pp. 440–454

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

size. For LOGCFL, a similar and simpler characterization is known by Venkateswaran [10] as the class of polynomial size Boolean circuits having polynomial proof tree size.

Although these two characterizations have a similar nature, we encounter with a slightly different circumstance when axiomatizing them. Since LOGDCFL is a deterministic class, there exists a unique witness, namely a proof tree for an accepting input to a circuit, whereas, there may exist more than one polynomial size proof trees for a general circuit which witnesses LOGCFL predicates. Such a difference causes a difficulty in the witnessing argument of the corresponding theories. However, we show that a slight modification of the argument of Cook and Nguyen implies witnessing theorems for both theories.

We also show that our theory $\mathbf{V}^{1}_{\mathrm{LCFL}}$ for LOGCFL is finitely axiomatizable by formalizing LOGCFL complete problems.

Cook proposed a research program called Bounded Reverse Mathematics whose aim is to decide how much computational concepts are needed to prove mathematical theorems. This is achieved by deciding which theorems of mathematics are provable in a given bounded arithmetic theory (See [3] for an exposition). For this direction, we show that $\mathbf{V}^{1}_{\mathrm{LCFL}}$ proves a form of the pumping lemma for context-free languages.

The essential part of formalizing the standard proof of the pumping lemma in $\mathbf{V}^{1}_{\mathrm{LCFL}}$ is to translate parse trees into proof trees of Boolean circuits so that we can argue about context-free languages in terms of circuits.

By carefully examining the textbook style proof of the pumping lemma, it turns out that proof also uses combinatorial arguments including the pigeonhole principle and PATH problem which is NL complete. Moreover, the inductive argument in the proof can be expressed as a number induction for LOGCFL decidable predicate.

## 2 Preliminaries

Throughout the paper we deal with two-sort theories and complexity classes. So first we briefly review their basic notions.

Two-sort logic uses two types of variables; number variables denoted by lower case letters $x, y, z \cdots$ represent natural numbers while string variables denoted by upper case letters $X, Y, Z \cdots$ represent binary strings.

### 2.1 Two-sort complexity classes

A two-sort function is a function with two sort of arguments $\bar{x}$ and $\bar{X}$. A function $F(\bar{x}, \bar{X})$ is a string function if its range is in strings. A function $f(\bar{x}, \bar{X})$ is a number function if its range is in numbers. As usual, we denote strings functions by uppercase letters and number functions by lower case letter. Predicates are identified with 0-1 valued number functions.

In the two-sort formalization, number variables play subsidiary roles, so in defining complexity classes, their measures are with respect to the length of string parameters.

Let $R(x_1, \ldots, x_k, X_1, \ldots, X_n)$ be a two-sort relation. As in [3], number variables $x_1, \ldots, x_k$ are presented in unary and string variables $X_1, \ldots, X_n$ are in binary form.

We define two sort complexity classes LOGCFL and LOGDCFL as follows:

▶ **Definition 1.** LOGCFL is the class of predicates which are logspace reducible to some context-free languages. LOGDCFL is the class of predicates which are logspace reducible to some deterministic context-free languages.

## 2.2 Two-sort theories

The language $L_2$ of two sort theories contains the symbols $Z(x)$, $x+y$, $x \cdot y$, $|X|$, $x=y$ and $x \in Y$. We also write $x \in Y$ as $Y(x)$. We use two sort of quantifiers; number quantifiers $(\forall x)$ and $(\exists x)$ and string quantifiers $(\forall X)$ and $(\exists X)$. Bounded number quantifiers are of the form $(\forall x < t)$ and $(\exists x < t)$. String bounded quantifiers are of the form

$$(\forall X < t)\varphi(X) \equiv (\forall X)(|X| < t \to \varphi(X)), \text{ and } (\exists X < t)\varphi(X) \equiv (\exists X)(|X| < t \wedge \varphi(X)).$$

An $L_2$ formula is called bounded if all its quantifiers are either bounded number quantifiers or bounded string quantifiers. The class $\Sigma_0^B$ consists of $L_2$ formulae whose quantifiers are bounded quantifiers only. The class $\Sigma_1^B$ consists of $L_2$ formulae whose quantifiers are bounded quantifiers, positive occurrences of existential bounded string quantifiers, and negative occurrences of universal bounded string quantifiers.

Our base theory is the following theory:

▶ **Definition 2.** The $L_2$ theory $V^0$ consists of the following axioms:
- $BASIC_2$ : finite number of defining axioms for symbols in $L_2$,
- extensionality axiom : $X = Y \leftrightarrow (|X| = |Y| \wedge (\forall i < |X|)(X(i) \leftrightarrow Y(i))$,
- $\Sigma_0^B$-COMP : $(\exists X)(\forall y < t)(X(y) \leftrightarrow \varphi(y))$, where $\varphi \in \Sigma_0^B$ does not contain $X$ as a free variable.

▶ **Definition 3.** A function $F(\bar{x}, \bar{X})$ is $\Sigma_1^B$ definable in a $L_2$-theory $T$ if there exists a $\Sigma_1^B$ formula $\varphi(\bar{x}, \bar{X}, Y)$ such that
- $T \vdash (\forall \bar{x})(\forall \bar{X})(\exists! Y)\varphi(\bar{x}, \bar{X}, Y)$, and
- $Y = F(\bar{x}, \bar{X}) \leftrightarrow \varphi(\bar{x}, \bar{X}, Y)$ holds in the standard model.

▶ **Theorem 4.** *A function is $\Sigma_1^B$ definable in $V^0$ if and only if it is computable in $AC^0$.*

The following property is also well-known and useful:

▶ **Theorem 5.** *Let $F$ be a $\Sigma_1^B$ definable function of $V^0$ and $V^0(F)$ be the theory $V^0$ extended by the function symbol for $F$ together with its defining axioms. Then $V^0(F)$ is an conservative extension of $V^0$.*

So we can use any $\Sigma_1^B$ definable function in $V^0$ without increasing its strength. In particular, the following functions are $\Sigma_1^B$ definable in $V^0$ which will be used in elsewhere in this paper:
- The number pairing function : $\langle x, y \rangle = \frac{(x+y+1)(x+y)}{2}$.
- The number $\mu$-operator :

$$\mu_{z<y}\varphi(x) = \begin{cases} \text{the least } x < y \text{ with } \varphi(x) & \text{if exists,} \\ 0 & \text{otherwise.} \end{cases}$$

- The value of $X$ at $y$ : $(X)^y = \mu_{y<a} X(\langle y, z \rangle)$ for $|X| < \langle a, b \rangle$.
- The row function : $Z^{[x]}(i) \leftrightarrow (i < |Z| \wedge Z(x, i))$.

The main tool of this paper is the method for constructing theories for subclasses of $P$ using complete problems which is developed by Cook and Nguyen. Details of this method can be found in [3], so we briefly overview the argument.

Suppose a function $F$ has a $\Sigma_0^B$ graph as

$$Y = F(X) \leftrightarrow (|Y| \leq t \wedge \delta_F(X < Y))$$

for some $L_2$ term $t$ and $\Sigma_0^B$ formula $\delta_F$. Suppose further that $V^0$ proves the uniqueness of the value of $F$. Let C be the class of relations which are $AC^0$ reducible to $F$. In this case, we define a theory for C as follows:

▶ **Definition 6.** The $L_2$-theory **VC** is axiomatized by the axioms of $V^0$ and the following axiom:

$$(\exists Y \leq b)(\forall i < b)\delta_F(X^{[i]}, Y^{[i]}).$$

Below we define theories with defining axiom for the function $F$ as above and the equivalence to the theory **VC** is ensured by showing that the aggregate function is $\Sigma_1^B$ definable within the theory.

▶ **Definition 7** (Aggregate Function)**.** Suppose that $F(\bar{x}, \bar{X})$ is such that for some $L_2$ term $t$

$$|F(\bar{x}, \bar{X})| \leq t(\bar{x}, \bar{X}).$$

Then $F^*(b, \bar{Z}, \bar{X})$ is a function such that

$$|F^*(b, \bar{Z}, \bar{X})| \leq \langle b, t(|\bar{Z}|, \bar{X})\rangle$$

and

$$F^*(b, \bar{Z}, \bar{X})(w) \leftrightarrow (\exists i < b)(\exists v < t)(w = \langle i, v\rangle \wedge F^*(b, (Z_1)^i, \ldots, (Z_k)^i, X_1^{[i]}, \ldots, X_n^{[i]})(v).$$

Finally, a direct connection between **VC** and FC is established:

▶ **Theorem 8** (Cook-Nguyen)**.** *A function is $\Sigma_1^B$ definable in* **VC** *if and only if it is in* FC.

We also use universal conservative extension $\widehat{\mathbf{VC}}$ of **VC** which is defined as follows: let $\overline{V^0}$ be the theory $V^0$ extended by function symbols for all $AC^0$ functions together with their defining axioms. Let $\delta'_F(X, Y)$ be the quantifier free defining axiom for $F$ which is equivalent to $\delta_F(X, Y)$ in $\overline{V^0}$ and define the function

$$Y = F'(X) \leftrightarrow (|Y| \leq t \wedge \delta'_F(X, Y)). \quad (*)$$

▶ **Definition 9.** $\widehat{\mathbf{VC}}$ is the universal theory over language of $\overline{V^0}$ plus $F'$ whose axioms are those for $\overline{V^0}$ and $(*)$.

**Remark.** Our theories defined below are slightly different from **VC** in that we add axiom scheme rather than a single axiom. Nevertheless, the witnessing and definability works by an almost similar argument.

## 3 A theory for LOGDCFL

We will define the theory $\mathbf{V}^{\text{LDCFL}}$ by formalizing the concept of polynomial proof tree of polynomial size multiplex-select circuits.

A multiplex select circuit is a circuit whose only gates are multiplex select gates. Inputs to a multiplex select gate are grouped into a bundle of $k \in O(\log n)$ steering bits and $2^k$ equal size bundles of $l \in O(\log n)$ data bits. The gate outputs the value of the data input bundle $d_i$ if the value of the steering input bundle is the binary expression of $i$. For detailed definition, see [7].

A proof tree of a circuit $C$ on input $X$ is a tree $PT(C, X)$ define inductively as follows:

- if $C$ consists of a single gate then $PT(C, X)$ consists of its steering input bundle and the data bundle selected by the steering input on $X$.
- if $C$ consists of more than two gates and $g$ is the output gate of $C$ then $PT(C, X)$ is defined as the tree rooted at $g$ with subtrees $PT(C_s, X)$ and $PT(C_d, X)$ where $C_s$ is the subcircuit with output at the steering input of $g$ and $C_d$ is the subcircuit with output at the data input selected by $s$ on $X$.

Note that the computation of multiplex select circuits are deterministic in the sense that there exists an unique proof tree for each input.

McKenzie et.al. [7] showed that multiplex select circuits characterize the class LOGDCFL.

▶ **Theorem 10** (McKenzie et.al.). *A predicate is in* LOGDCFL *if and only if it is decidable by $AC^0$-uniform poly-size family of multiplex select circuits with polynomial proof tree size.*

We code a multiplex select gate by a triple $\langle g, c, k\rangle$ where $g$ is the index of its steering input, $k$ is the number of data inputs, and $c + j$ is the index of $j$-th data input for $j \leq k$.

As usual, a circuit is coded by a two-dimensional array but with semantics which differs from that of Boolean circuits. Recall that a non-input gate in a multiplex-select circuit has two types of inputs; one steering input and $2^l$ data inputs where $l$ is the length of the steering input bundle. So we define $AC^0$ functions which specifies these inputs.

Let $E$ be a two-dimensional array with $|E| = \langle n, n\rangle$. A gate $x < n$ with no input is regarded as an input data bundle with the data $x$. Otherwise, $x$ is a non-input gate and we define its steering input $s(x, E)$ and the first data input $c(x, E)$ as follows:

$$s(x, E) = \mu y < x E(x, y),$$

$$c(x, E) = \begin{cases} \mu y < x(E(x, y) \wedge s(x, E)) < y & \text{if it exists,} \\ s(x, E) & \text{otherwise.} \end{cases}$$

We omit the parameter $E$ if it is clear from the context.

The output of a multiplex select gate is given by the following rule:

> for a gate $x$, if it receives an input $j$ from its steering input $s(x)$ then it outputs the input from the data input of index $\min(c(x) + j, x - 1)$.

We use the notion of degrees of gates in a multiplex-select circuit which is defined as:

▶ **Definition 11.** Let $E$ be a circuit such that $|E| = \langle n, n\rangle$ and $x < n$. We define the degree $\deg(x, E)$ recursively as

$$\deg(x, E) = \begin{cases} 1 & \text{if } x \text{ is an input gate,} \\ \deg(s(x)) + \deg(\min(c(x) + j, x - 1)) & \\ \quad \text{if } x \text{ is an non-input and } s(x) \text{ outputs } j. & \end{cases}$$

It is not difficult to see that a circuit family has polynomial size proof trees if and only if it has polynomial degrees.

We define an axiom expressing the concept of multiplex-select circuits with polynomial degrees. The idea is to code the computation of a given circuit by a string $Z$ such that a

- if $degree(x)$ is bounded by a given polynomial $p(n)$ then $(Z)^x = \langle v_x, \deg(x)\rangle$ where $v_x$ is the output of $x$, and
- otherwise $(Z)^x = \langle n, p(n)\rangle$.

Formally, define the $L_2$ formula $MSCDeg(n, x, E, Z)$ as the conjunction of the following formulae:

- $x$ is an input gate: $(\forall y < x)\neg E(x, y) \to (Z)^x = \langle x, 1\rangle$,
- $x$ is a non-input gate having a degree less than $p(n)$:

$$(\exists y < x)E(x, y) \to (Z)_1^{s(x)} + (Z)_1^{\min(c(x)+(Z)_0^{s(x)}, x-1)} < p(n) - 1 \wedge$$
$$(Z)^x = \langle (Z)_0^{\min(c(x)+(Z)_0^{s(x)}, x-1)}, (Z)_1^{s(x)} + (Z)_1^{\min(c(x)+(Z)_0^{s(x)}, x-1)}\rangle,$$

- $x$ is an non-input gate having a degree greater than $p(n) - 1$:

$$(\exists y < x)E(x, y) \to (Z)_1^{s(x)} + (Z)_1^{\min(c(x)+(Z)_0^{s(x)}, x-1)} \geq p(n) \wedge (Z)^x = \langle n, p(n)\rangle.$$

For a $L_2$-term $p(n)$ we define the formula

$$MSCDeg\text{-}COMP_p : (\forall n)(\forall E < \langle n, n\rangle)(\exists Z\langle n, \langle n, p(n)\rangle\rangle)(\forall x < n)MSCDeg(n, x, E, Z).$$

▶ **Definition 12.** The $L_2$-theory $\mathbf{V}^0\mathrm{LDCFL}$ is $V^0$ extended by $MSCDeg\text{-}COMP_p$ for each $p \in L_2\}$.

**Remark.** We have made several simplifications in the formalization of multiplex-select circuits which is still general enough to capture the original ones in the sense that we can effectively compute a code $E(C, X)$ from the original circuit code $C$ and an input $X$. Firstly, we can convert an $O(\log |X|)$ input bit bundle into a number representing a gate index using $AC^0$ function. The original definition allows each bundle to be extended by constant bits which also can be computed in $AC^0$.

Finally, the restriction of date inputs bit to consecutive ones as $c(x), \ldots c(x) + j$ can express the original circuit as follows. Let $d$ be the $i$-th data input to $x$. We introduce a "copy" gate $C_d$ of $d$ having a single bit steering input and a single data input from $d$. Then we let the gate position of $C_d$ to be $c(x) + i$.

▶ **Theorem 13.** *A function is $\Sigma_1^B$ definable in $\mathbf{V}^0\mathrm{LDCFL}$ if and only if it is* LOGDCFL*-computable.*

To prove Theorem 13, we use the formalization as in Cook-Nguyen's Book [3]. First we show the uniqueness of proof trees.

▶ **Lemma 14.** $\mathbf{V}^0\mathrm{LDCFL}$ *proves the following:*

$$(\forall Z_0, Z_1 < n)[((\forall x < n)MSCDeg_p(n, x, E, Z_0) \wedge (\forall x < n)MSCDeg_p(n, x, E, Z_1)) \to Z_0 = Z_1].$$

(Proof). Since $MSCDeg_p$ is a $\Sigma_0^B$ formula, this follows from an $\Sigma_0^B$-IND instance which is available in $V^0$. The most essential part is to show that the aggregate function for $MSCDeg_p$ is $\Sigma_1^B$-definable in $\mathbf{V}^0\mathrm{LDCFL}$.

▶ **Lemma 15.** *The aggregate function for $MSCDeg\text{-}COMP_p$ defined by the formula*

$$(\forall n)(\forall E < \langle\langle n, n\rangle, b\rangle)(\exists Y)(\forall i < b)(\forall x < n)MSCDeg_p(n, x, E^{[i]}, Y^{[i]})$$

*is $\Sigma_1^B$-definable in $\mathbf{V}^0\mathrm{LDCFL}$.*

The proof is more or less identical to that for $VP$ (Lemma VIII 1.10 in [3]).

Now, Theorem 13 follows from the following:

▶ **Lemma 16.** *The $FAC^0$ closure of functions $F_p$ with defining axiom*

$$Y = F_p(n, E) \Leftrightarrow |Y| \leq t_p \wedge (\forall x < n)MSCDeg_p(n, x, E, Y)$$

*for $p \in L_2$, denoted by $FAC^0(DegP)$, is identical to $\mathcal{F}\mathrm{LOGDCFL}$.*

(Proof). To show that $FAC^0(DegP) \subseteq \mathcal{F}_{\text{LOGDCFL}}$ we remark that $F_p \in \mathcal{F}_{\text{LOGDCFL}}$ for all $p \in L_2$.

For the converse inclusion, it suffices to show that any bit in $F \in \mathcal{F}_{\text{LOGDCFL}}$ can be computed using some $F_p$ together with $AC^0$ operations.

Using the technique developed by Cook and Nguyen [3], we can prove Theorem 13 from Lemmata 14, 15 and 16.

Since LOGDCFL contains L, we expect that the same inclusion holds for corresponding theories. The theory **VL** is axiomatized by $V^0$ together with the following axiom:

$$PATH \equiv Unique(a, E) \rightarrow (\exists P \leq \langle a, a \rangle)\delta_{PATH}(a, E, P),$$

where

$$Unique(a, E) \equiv (\forall x < a)(\exists! y < a)E(x, y),$$
$$\delta_{PATH}(a, E, P) \equiv (P)^0 = 0 \wedge (\forall v < a)(E((P)^v, (P)^{v+1}) \wedge (P)^{v+1} < a).$$

▶ **Theorem 17.** $\mathbf{V}_{\text{LDCFL}}$ *contains* **VL**.

(Proof). It suffices to show that $\mathbf{V}_{\text{LDCFL}}$ proves $PATH$. Let $E$ satisfy $Unique(a, E)$. For each $x < a$, we denote the unique $y$ such that $E(x, y)$ by $d(x)$. We first design a multiplex select gate such that if the steering input is $x$ then it outputs $d(x)$. This gate is easily constructed by setting its data inputs as $d(0), \ldots, d(a-1)$. Then we connect copies of this gate in a sequential manner, that is the first gate receives its steering input from $x$, the second gate from the first gate, and so on. Furthermore, all gates receive data inputs from $d(0), \ldots, d(a-1)$.

Suppose a node $b < n$ is reachable from $a$ by a path of length $k \leq n$. Then it is easy to see that there exists a gate in this circuit with label $\langle b, 2k+1 \rangle$.

The above argument can be formalized in $\mathbf{V}_{\text{LDCFL}}$, that is, for $p(n) \leq 2n+1$, we have

$$(\exists Z)(\forall x < n)MSCDeg_p(n, x, E', Z)$$

where $E'$ is the code for the above circuit. Furthermore, the witness $P$ in $PATH$ can be constructed from $Z$ using $\Sigma_0^B$-COMP.

## 4 A theory for LOGCFL

We now turn to show that the axiomatization of proof tree size concept also captures the class LOGCFL. First we recall the following circuit characterization of the class:

▶ **Theorem 18** (Venkateswaran [10])**.** LOGCFL *is the class of predicates which are computable by polynomial size circuits with polynomial proof trees. This equivalence is true even if we restrict circuits to semi-unbounded fan-in.*

(Proof). The former statement is due to [10]. So we prove the latter part.

It is readily seen that a circuit can be transformed to an equivalent semi-unbounded fan-in circuit with polynomial increase in size by the transformation of unbounded fan-in AND gates by a fan-in two subcircuits of $O(\log n)$ depth. It remains to show that the degree of the resulting circuit also has polynomial increase. For this, it suffices to show that the replacement of unbounded fan-in AND-gates by subcircuits yields polynomial increase in the degree,that is the number of nodes in a proof tree.

Let $g$ be an AND-gate in the original circuit having inputs from $h_0, \ldots, h_{k-1}$ and $g'$ be the gate on the top of the subcircuit which is a substitute for $g$. It is not difficult to see that $\deg(g') = \deg(g) + k$. More strictly, this is proved by induction on the depth of gates.

The advantage of the semi-unbounded fan-in restriction is that we do not need vector summation to compute degrees. Thus we can axiomatize our theory based on $V^0$ rather than $\mathbf{VTC^0}$.

Now we define an axiom expressing polynomial degrees in a similar manner as **VP** of Cook-Nguyen [3]. First we will give an intuitive idea: Let $E$ be a two-dimensional array coding a circuit. The input-output relation of $E$ differs from that for multiplex select circuits.

Let $G(x)$ be an unary predicate determining the type of a gate $x$ so that $x$ is an AND gate if $G(x)$ and an OR gate otherwise. If $G(x)$ holds then $x$ receives inputs from two gates defined by

$$c_0(x) = \mu y < x E(x, y),$$
$$c_1(x) = \mu y < x (E(x, y) \wedge c_0(x) < y).$$

If $\neg G(x)$ then $x$ receives inputs from all $y < x$ such that $E(x, y)$.

The computation of a circuit given by $E$ and $G$ as above is coded by a string $Z$ such that

$$(Z)^0 = p(n),\ (Z)^1 = 1,$$
$$(Z)^x = \begin{cases} \deg(x) & \text{if } x \text{ outputs 1 and } \deg(x) < p(n), \\ p(n) & \text{otherwise} \end{cases}$$

for $x \geq 2$.

Putting these altogether, we formally define

$$\begin{aligned}
&MCVDeg_p(n, E, G, Z) \Leftrightarrow \\
&(Z)^0 = p(n) \wedge (Z)^1 = 1 \wedge \\
&(\forall x < n)(x \geq 2 \rightarrow \\
&(G(x) \wedge (((Z)^{c_0(x)} + (Z)^{c_1(x)} < p(n) - 1 \wedge (Z)^x = (Z)^{c_0(x)} + (Z)^{c_1(x)} + 1) \vee \\
&((Z)^{c_0(x)} + (Z)^{c_1(x)} \geq p(n) - 1 \wedge (Z)^x = p(n))) \vee \\
&(\neg G(x) \wedge ((\exists y < x)((Z)^y < p(n) - 1 \wedge E(x, y) \wedge (Z)^x = (Z)^y + 1) \vee \\
&((\forall y < x)(E(x, y) \rightarrow (Z)^y \geq p(n) - 1 \wedge (Z)^x = p(n))))).
\end{aligned}$$

where $p \in L_2$.

▶ **Definition 19.** We define the $L_2$-theory $\mathbf{V}_{\mathrm{LCFL}}$ as

$$V^0 + \{(\forall n)(\forall E < \langle n, n\rangle)(\forall G < n)(\exists Z < \langle n, p(n)\rangle) MCVDeg_p(n, E, G, Z) : p \in L_2\}.$$

It is possible to define our theory using $SAC^1$ circuits which can be formalized by a single axiom. Nevertheless, we choose the above axiomatization as it is easier to formalize other computational concepts within the theory. First we show that our theories preserve the inclusion relation of corresponding complexity classes:

▶ **Theorem 20.** $\mathbf{V}_{\mathrm{LCFL}}$ *proves* $CONN$*, thus it contains* **VNL**.

(Proof). We argue in $\mathbf{V}_{\mathrm{LCFL}}$. Let $E < \langle n, n\rangle$ be given. We need to show that

$$(\exists Y < \langle a, a\rangle)(\delta_{CONN}(a, E, Y) \wedge Y(a, 1)).$$

To this end, we construct a graph $C < \langle\langle a, a\rangle, \langle a, a\rangle\rangle$ such that

$$C(0, x, 1, y) \leftrightarrow x = 0 \wedge E(0, y)$$

and

$$C(n, x, m, y) \leftrightarrow m = n + 1 \wedge E(x, y)$$

hold. That is, $C$ has layers such that each column contains copies of all nodes in $E$. The edge relation of $C$ is given so that the node $x$ in $i$-th layer has an edge to the node $y$ in $(i+1)$-st layer if $E(x,y)$ holds for $i > 0$. Furthermore, on 0-th layer, only the node 0 has an edge to the node $y$ in the first layer if $E(0,y)$.

Now we show that a small modification of $C$ yields a circuit deciding the axiom CONN for **VNL**. In the new circuit $C'$, we have the first column containing 0 and 1, say $\langle 0,0\rangle$ and $\langle 1,0\rangle$, and the number of columns of $C$ are incremented by 1 in $C'$. Thus the gate $\langle i,j\rangle$ in $C$ corresponds to $\langle i,j+1\rangle$ in $C'$. Thus we define

$$C'(n,x,m,y) \leftrightarrow (n=0 \wedge x=1 \wedge m=1 \wedge y=0) \vee (C(n-1,x,m-1,y) \wedge n>0 \wedge m>0).$$

Furthermore, we let $G$ be such that $(\forall n)(\forall x)G(n,x)$, that is, all gates in $C'$ are AND gates.

Now let $p(n) = n$. Then we have $(\exists Z)MCVDeg_p(n,C',G,Z)$ and we have an $AC^0$ function $F$ such that

$$MCVDeg_p(n,C',G,Z) \rightarrow \delta_{CONN}(n,E,F(Z)).$$

Also note that a proof tree of a gate $\langle i,x\rangle$ is a path from $\langle 0,1\rangle$ to $\langle i,x\rangle$. Thus we are done.

▶ **Theorem 21.** **V**LCFL *contains* **V**LDCFL.

(Proof). It suffices to show that **V**LCFL proves $MSCDeg\text{-}COMP_p$ for any $p \in L_2$. This is achieved by giving circuits simulating multiplex-select circuits which have polynomial size and polynomial proof size. Let $g$ be an MS-gate with the steering input $s$ and data inputs $d_0,\ldots,d_l$. The Boolean circuit $C_g$ simulating $g$ can be obtained by building the following subcircuits:

- $C^i_{sel}$ outputs 1 if the steering input $s$ selects the data input $d_i$, and
- $C^j_{out}$ outputs the bits of the $j$-th output bit of $g$.

Then $C_g$ is defined as

$$\bigvee_{1\le i\le l} (C^i_{sel} \wedge C^i_{out}).$$

The subcircuit $C^i_{sel}$ is defined as $\bigwedge_{1\le j\le l} l_j$ where

$$l_i = \begin{cases} s_j & \text{if the } j\text{-th bit of } i \text{ is 1} \\ \neg s_j & \text{otherwise.} \end{cases}$$

Then $C^j_{out}$ is computed by the circuit

$$\bigvee_{1\le i\le l} (C^i_{sel} \wedge d^j_i).$$

It is not difficult to see that the above construction can be described by $\Sigma^B_0$ relation. So we have the $\Sigma^B_0$ description of the translation of the whole MS-circuit $C$ by an equivalent Boolean circuit.

Although the above construction does not yields semi-unbounded fan-in circuits, it can be converted into an equivalent $SAC^1$ circuit.

It is also true that **V**LCFL is contained in the theory for $AC^1$ which is axiomatized by $V^0$ and the axiom

$$(\exists Y \le \langle |n|+1, n\rangle \delta_{LMCV}(n,|n|,E,G,I,Y)$$

where

$$\begin{aligned} \delta_{LMCV}(n,d,E,G,I,Y) \equiv\ & (\forall x<n)(\forall z<d)((Y(0,x) \leftrightarrow I(x)) \wedge \\ (Y(z+1,x) \leftrightarrow\ & (G(z+1,x) \wedge (\forall u<n)(E(z,u,x) \rightarrow Y(z,u))) \vee \\ & (\neg G(z+1,x) \wedge (\exists u<n)(E(z,u,x) \wedge Y(z,u)))). \end{aligned}$$

▶ **Theorem 22.** VLCFL *is contained in* $\mathbf{VAC^1}$.

Actually, we prove a stronger theorem stating that any polynomial size circuit family with polynomial proof tree size has an equivalent $SAC^1$ circuits.

▶ **Theorem 23.** *Let* $p, q \in L_2$. $\mathbf{VTC^0}$ *proves that for any circuit* $C$ *with size* $p$ *and proof tree size* $q$ *there exists a semi-unbounded circuit* $C'$ *such that for any input* $X$,

$$C \text{ accepts } X \text{ in proof tree size } q \Leftrightarrow C' \text{ accepts } X.$$

(Proof). We use the idea similar to realizable pairs of Ruzzo [8]. Let $g$ be a gate of $C$ and $\Gamma$ be a set of nodes in $C$. W define $C_{g,\Gamma}$ to be a directed acyclic subgraph of $C$ whose root is $g$ and sinks are $\Gamma$. We say that $C_{g,\Gamma}$ is realizable within size $p$ and proof tree size $q$ if $size(C_{g,\Gamma}) \leq p$ and $g$ has proof tree size $\leq q$ provided that all nonleaves of $\Gamma$ are assigned the value 1.

It suffices to construct a semi-unbounded fan-in circuit deciding whether $C_{g,\Gamma}$ is realizable with size $p$ and proof tree size $q$ since $C = C_{g_0,\emptyset}$ is realizable if and only if $C$ accepts the input where $g_0$ is the output gate of $C$. So we construct a recursive procedure $realize(C, p, q)$ which works as follows:

In each recursive step, $realize(C, p, q)$ splits $C$ into two parts $C_0$ and $C_1$ each having approximately half size of $C$ and recursively check whether $realize(C_0, p/2, i)$ and $realize(C_1, p/2, j)$ where $i + j = q$. This procedure is given as follows:

**procedure** realize(C,p,q);
**begin**
**if** $C$ consists of a single gate $g$ **then if** $g$ has the value 1 and $p, q \geq 1$
**then** ACCEPT **else** REJECT
**else**
guess $i$ with $0 < i < q$ and a gate $s$ in $C$;
$C_0$ := subcircuit of $C$ rooted at $s$; $C_1 := C \setminus C_0$ with $s$ replaced by 1;
check in parallel
$realize(C_0, p/2, i)$ AND $realize(C_1, p/2, q - i)$
**end**

We will show that this algorithm can be converted into logarithmic depth semi-unbounded fan-in circuits.

First we remark that checking whether $size(C) < p$ can be done by an $NC^1$ circuit using threshold circuits.

Since the number of the nestings of recursive calls is logarithmic in $p$, the total number of subcircuits $C_0$s and $C_1$s created along the execution of the procedure is bounded by polynomial. Thus the number of gates used in checking circuit sizes is polynomially bounded.

Now each recursive steps can be expressed by an AND of $size(C) < p$ and the OR of the following subroutines:

- the base case with a single node $g$,
- $realize(C_0, q/2, i)$ AND $realize(C_1, p/2, q - i)$ for $0 < i < q$ and all nodes in $C$.

The fan-in of this $OR$ gate is $O(p \cdot q)$ which is polynomial. Furthermore, all AND gates required is bounded fan-in.

By recalling that the number of the nestings of recursive calls is $\log p = O(\log n)$, we conclude that the procedure $realize$ can be transformed into an $SAC^1$ circuit family.

It is not hard, though tedious, to show that the above construction can be demonstrated by $\Sigma_0^B$ relation. To check the correctness of the construction, we need counting argument for the evaluation of circuit sizes. So the whole argument can be formalized in $\mathbf{VTC^0}$.

▶ **Corollary 24.** $\mathbf{V}$LCFL *and* $V$-$Q^{SAC}$ *proves the same* $L_2$ *sentences.*

Now we show that $\mathbf{V}$LCFL corresponds to LOGCFL.

▶ **Theorem 25.** *A function is* $\Sigma_1^B$ *definable in* $\mathbf{V}$LCFL *if and only if it is in* $\mathcal{F}$LOGCFL.

Note that the witness $Z$ for $MCVDeg_p$ axiom is not always optimal in a sense that the degree of the output of $Z$ is larger than $p(n)$ while there is another witness $Z'$ whose output degree is less than or equal to $p(n)$. Nevertheless, we can avoid such undesirable cases by considering $SAC^1$ circuits.

▶ **Theorem 26** (Definability for $\mathbf{V}$LCFL)**.** *If a function of polynomial growth is bitwise computable in* LOGCFL *then it is* $\Sigma_1^B$ *definable in* $\mathbf{V}$LCFL.

(Proof Sketch). In [6], it is shown that $V$-$Q^{SAC}$ proves that $SAC^1$ is closed under complementation. By Corollary 24, this is also provable in $\mathbf{V}$LCFL.

Moreover, any $SAC^1$ circuit $C$ is a polynomial size circuit having polynomial proof tree size such that for any input $X$ and any $Z$ witnessing $MCVP$ axiom for $C$, $C$ outputs true if and only if the degree of the output gate is less than $p(|X|)$ for a given proof tree bound $p$.

Let $F(X)$ be a LOGCFL function with polynomial growth. To show that $\mathbf{V}$LCFL $\Sigma_1^B$ defines $F$, it suffices to show that circuits can be combined into a multi output circuit. This can be done using circuits and their complementary circuits which is also an $SAC^1$ circuit provably in $\mathbf{V}$LCFL.

For witnessing we use the argument using the conservative universal extension $\widehat{\mathbf{V}\text{LCFL}}$ of $\mathbf{V}$LCFL.

▶ **Theorem 27** (Witnessing for $\mathbf{V}$LCFL)**.** *If a function is* $\Sigma_1^B$ *definable in* $\mathbf{V}$LCFL *then it is of polynomial growth and bitwise computable in* LOGCFL.

(Proof). We define the universal conservative extension $\widehat{\mathbf{V}\text{LCFL}}$ by introducing for each $p \in L_2$ a function symbol $F_p$ such that

$$F_p(n, E, G) = Z \leftrightarrow MCVDeg_p(n, E, G, Z).$$

Using Herbrand-style argument, we can show that provably total functions of $\widehat{\mathbf{V}\text{LCFL}}$ are in the $AC^0$ closure of $\{F_p : p \in L_2\}$. As LOGCFL is closed under $AC^0$ operations, it suffices to show that $F_p$ is witnessed by a LOGCFL algorithm. In order to compute $F_p$ we modify NAuxPDA machine simulating polynomial size circuits with polynomial proof trees as in [7]. The machine makes an depth-first search through a circuit and at each step, it computes the degree of the currently visiting gate using the work tape. This can be done in logarithmic space by expressing degrees in binary. Therefore, this gives an NAuxPDA computing $F_p$.

## 5 Finite axiomatizability

In this section we show that the theory $\mathbf{V}$LCFL is finitely axiomatizable. The idea is to formalize LOGCFL complete problems such as acyclic conjunctive query problem (Gottlob et.al. [4]) or word problem for finite groupoids (Bedard et.al. [1]).

First we modify the theory $\mathbf{V}^{\mathrm{LCFL}}$ so that it can argue about circuit families which have $\Sigma_0^B$ direct connection languages. For an $L_2$-formula $\varphi(x,y)$ we define the string

$$E_\varphi(n)(x,y) \Leftrightarrow x < n \wedge y < n \wedge \varphi(x,y).$$

Similarly for an $L_2$-formula $\psi(x)$ we define the string

$$G_\psi(n)(x) \Leftrightarrow x < n \wedge \psi(x).$$

Then we define the theory

$$\mathbf{V}^{\mathrm{LCFL}\prime} \equiv V^0 + \{(\forall n)(\exists Z < \langle n, p(n)\rangle MCVDeg_p(n, E_\varphi(n), G_\psi(n), Z) : \varphi, \psi \in \Sigma_0^B, p \in L_2\}.$$

It is not difficult to see that

▶ **Lemma 28.** $\mathbf{V}^{\mathrm{LCFL}\prime} = \mathbf{V}^{\mathrm{LCFL}}$.

Using $\mathbf{V}^{\mathrm{LCFL}\prime}$ we have

▶ **Theorem 29.** $\mathbf{V}^{\mathrm{LCFL}\prime}$ *and hence* $\mathbf{V}^{\mathrm{LCFL}}$ *is finitely axiomatizable.*

(Proof Sketch). We can formalize the acyclic conjunctive query problem in $L_2$ and define the direct connection language $\varphi$ and $\psi$ of the polynomial size circuit family deciding it in proof tree size $p(n)$. Then in $\mathbf{V}^{\mathrm{LCFL}\prime}$ we can show that any circuit family with polynomial proof tree size can be reduced to it. Thus $\mathbf{V}^{\mathrm{LCFL}\prime}$ is finitely axiomatizable.
**Remark.** Although it is likely that the same argument holds for $\mathbf{V}^{\mathrm{LDCFL}}$, we do not know whether there exists a complete problem for LOGDCFL.

## 6 Provability of the pumping lemma for CFLs

We now turn to the problem of the provability of the pumping lemma. Intuitively, the pumping lemma states that a sufficiently long word in a context-free language can be iterately "pumped up" to another word in the language. More precisely,

▶ **Theorem 30** (The pumping lemma for CFLs). *Let $G = (N, T, P, S)$ be a context-free grammar in Chomsky normal form. Suppose $W \in L(G)$ is such that $|W| \geq 2^{m-1} + 1$ where $m = |N|$. Then there exist $W_1, \ldots, W_5$ such that $W = W_1 \cdots W_5$, $|W_2 W_4| \geq 1$, $|W_2 W_3 W_4| \leq 2^m$ and $W_1 W_2^i W_3 W_4^i W_5 \in L(G)$ for all $i \in \omega$.*

In order to formalize Theorem 32, we need expressions which refer to the exponentiation like $y \geq 2^x$. Recall that the relation $y = 2^x$ is $\Delta_0$ definable in $I\Delta_0$ although the function is not total in the theory. Likewise, in $L_2$ we can define the exponentiation relation using a $\Sigma_0^B$ formula. In particular, the assumption $|W| \geq 2^{s-1} + 1$ in Definition 31 below is expressible in the language of $\mathbf{V}^{\mathrm{LCFL}}$.

Next we show how to code context-free grammars. Let $G = (N, T, P, S)$ be a CFG in Chomsky normal form. Since $N$ and $T$ are finite sets, We code it as $T = \{0, \ldots n-1\}$, $N = \{n, \ldots, n+s\}$, $S = n$. Recall that rules of context-free grammars in Chomsky normal form are either of the form $a \to bc$ or $a \to l$ for $a, b, c \in N$ and $l \in T$. We let $P \subseteq N \times (T \cup N^2)$ so that

$$P(a,x) \Leftrightarrow a \to x \text{ is a rule in } P.$$

We also write $a \to_P x$ instead of $P(a,x)$.

A parse tree $T$ of height $l$ is a layered binary tree such that each row $T^{[i]}$ is obtained from $T^{[i-1]}$ by applying rules in $P$ to nonterminal symbols simultaneously. Each symbol $T^{[i]}[k]$

is of the form $\langle n, k' \rangle$ denoting that $u \leq n + s$ is a symbol assigned to the slot and $k'$ is the index of its parent in $T^{[i-1]}$, where $X[y]$ is the value of $X$ at $y$ (denoted by $(X)^y$ in [3]).

More precisely, $T$ is a parse tree if the following conditions hold:

- $T^{[0]}$ consists of a single nonterminal symbol $a$ with $n \leq a \leq n + s$ and for all $i > 0$ and $k$, $T^{[i]}[k] = \langle u, k' \rangle$ for some $u \leq n + s$ and $k'$.
- If $T^{[i]}[k] = \langle u, k' \rangle$ with $u \geq n$ then $T^{[i]}[k + j] = \langle v, k' \rangle$ for $v \geq n$ and exactly one $j \in \{-1, 1\}$ such that $w \to_P uv$ for $w = (T^{[i-1]}[k'])_0$. Moreover, either one of the following holds:
  - there exists a unique $k''$ such that $T^{[i+1]}[k''] = \langle v, k \rangle$, $v < n$ and $u \to_P v$,
  - there exists a unique $k''$ such that $T^{[i+1]}[k''] = \langle v, k \rangle$, $T^{[i+1]}[k'' + 1] = \langle v', k \rangle$, $n \geq v, v' \geq n + s$ and $u \to_P vv'$.
- If $T^{[i]}[k] = \langle u, k' \rangle$ with $u < n$ then there exists a unique $k''$ such that $T^{[i+1]}[k''] = \langle v, k \rangle$ whenever $i \leq l$ where $l$ is the number of rows in $T$. In addition, one of $(T^{[i-1]}[k'])_0 = u$ or $(T^{[i-1]}[k'])_0 \to u$ holds.
- For any $i$ and $k < k'$ if $T^{[i+1]}[c]$ and $T^{[i+1]}[c]$ are children of $T^{[i]}[k]$ and $T^{[i]}[k']$ then $c < c'$.
- $(T^{[l]}[k]))_0 < n$ for all $k$.

Based on the above codings, it is easy, though tedious, to see that there exists a $\Sigma_0^B$ formula

$$Parse(a, l, n, s.P, T, X) \equiv$$
$$T \text{ is a parse tree of length } l \text{ starting from } a \text{ which generates } X.$$

▶ **Definition 31.** We define $PL(n, s, P)$ to be the following formula:

$$(\forall X)(\exists T)Parse(s, l, n, n, T, P, X) \wedge |X| \geq 2^{s-1} + 1$$
$$\to (\exists X_1, X_2, X_3, X_4, X_5)(X = X_1X_2X_3X_4X_5 \wedge |X_2X_4| \geq 1 \wedge |X_2X_3X_4| \leq 2^s \wedge$$
$$(\forall i)(\exists l')(\exists T')Parse(a, l', n, n, P, T', X_1X_2^iX_3X_4^iX_5)).$$

As previously stated, this is provable in **V**LCFL, namely,

▶ **Theorem 32.** **V**LCFL *proves* $(\forall n)(\forall s)(\forall P)PL(n, s, P)$.

The main idea of the proof is to simulate CFGs by circuits so that the transformation of parse trees as in the standard proof of the pumping lemma as in [5] can be interpreted in terms of proof tree. Let $(E, G)$ be a code of a circuit as in the axiom $MSCDeg_p$ but with $n$ input gates whose values are unspecified and $X$ be a string with $|X| = n$. Then we define $(E, G)$ with input $X$ by $(E_X, G_X)$. Our main tool is the following:

▶ **Lemma 33.** *There exists an $AC^0$ function $F(m, n, s, P) = (a, E, G)$ which is $\Sigma_1^B$ definable in $V^0$ and an $L_2$-term $r(n, s, P, X)$ such that $(E, G)$ is a code of a semi-unbounded fan-in circuit with $m$ unspecified input gates and* **V**LCFL *proves*

$$(\forall X)[(\exists T)Parse(s, l, n, n, P, T, X)$$
$$\leftrightarrow (\exists Z)(MCVP_r(a, E_X, G_X, Z) \wedge Z[a] < r(|X|, n, s, P))].$$

*Moreover, there exists an $AC^0$ function $T(Z)$ such that $V^0$ proves the following:*

$$(\forall X)[(MCVP_r(a, E_X, G_X, Z) \wedge Z[a] < r(|X|, n, s, P)) \to Parse(s, l, n, n, T(Z), P, X)].$$

(Proof Sketch). As in Example 1 in page 220 of Ruzzo [8], we construct a circuit checking whether an input string belongs to a given context-free language.

We construct a circuit which simulates a given CFG $G = (n, s, P)$ on input $X$. The idea is to introduce a gate which checks whether $a \Rightarrow^* x_i \cdots x_j$ for each nonterminal symbol $a$ and a substring $x_i \cdots x_j$ of $X$.

In particular, the output gate is an OR gate checking whether $n \Rightarrow^* x_1 \cdots x_n$ (Recall that the start symbol is coded by $n$). This is done by introducing an AND gate for each rule $n \to ab$ and a partition $x_1 \cdots x_i / x_{i+1} \cdots x_n$. This AND gate gives an output to the output OR gate and receives inputs from gates checking $a \Rightarrow^* x_1 \cdots x_i$ and $b \Rightarrow^* x_{i+1} \cdots x_n$.

Gates which check whether $a \Rightarrow^* x_i \cdots x_j$ are defined similarly. Furthermore, at the input level, we connect subcircuits deciding that a given binary string is a code of a terminal symbol.

It is easy to see that the edge relation of this circuit is $AC^0$ computable. Also it can be seen that it has polynomial size and polynomial proof size and semi-unbounded fan-in.

We also remark that the above construction of the circuit ensures that its proof tree corresponds to a parse tree of $n \Rightarrow^* X$, so it is easy to construct an $AC^0$ function transforming a proof tree into a parse tree.

(Proof of Theorem 32) It suffices to show that the transformation of parse trees in the standard proof of the pumping lemma can be paraphrased in terms of proof trees in the circuit $F(n, s, P, X)$ of Lemma 33.

First we note that by Lemma 33, it follows that the predicate $(\exists T)Parse(s, l, n, n, T, P, X)$ is equivalent to an open formula in the universal conservative extension $\widehat{\mathbf{V}^{\mathrm{LCFL}}}$. So in the following we argue in $\widehat{\mathbf{V}^{\mathrm{LCFL}}}$.

Let $X \in L(G)$ be such that $|X| \geq 2^{s-1} + 1$. First we claim that there exists a partition $X_1, \ldots, X_5$ of $X$ as in the proof of the pumping lemma.

By assumption, $(\exists T)Parse(s, l, n, n, P, T, X)$ holds for some $l$. So by Lemma 33 we have

$$(\exists Z)(MCVP_r(a, E_x, G_x, Z) \wedge Z[a] < r(m, n, s, P).$$

Since the circuit $(E_X, G_X)$ is semi-unbounded fan-in, we can extract a proof tree from $X$ which is binary branching.

**Claim.** There exists a path $R$ from the output to some input in $R$ which is longer than $s$.

(Proof of Claim). The construction of the circuit $(E_X, G_X)$ ensures that any proof tree contains all bits of $X$. So by a combinatorial argument formalized in $\mathbf{V}^{\mathrm{LCFL}}$ we have the claim.

From the above claim and the pigeonhole principle for $l(R) \to s$ where $l(R)$ is the length of the path $R$, we can choose two occurrences of some nonterminal symbol $c$.

In order to choose the position of such occurrences, we slightly modify the construction of the circuit in Lemma 33 so that the OR gate checking $a \Rightarrow^* x_i \cdots x_j$ is assigned a label $a$. Note that this modification is also $AC^0$ computable.

Let $X'$ and $X''$ be substrings of $X$ generated by the first and the second occurrences of $c$ respectively. We can execute these substrings using the PATH axiom. Moreover, the subtree starting from a given node is also computable by PATH.

Let $X_1, X_2, X_3, X_4, X_5$ be such that $X = X_1 X' X_5$, $X' = X_2 X'' X_4$ and $X_3 = X''$. Define

$$pump(X, i) = pump(X_1, X_2, X_3, X_4, X_5, i) = X_1 X_2^i X_3 X_4^i X_5.$$

It is easy to see that $pump$ is $\Sigma_1^B$ definable in $V^0$.

Now we show that $\widehat{\mathbf{V}^{\mathrm{LCFL}}}$ proves the following which we denote by $(\forall i)PT(i, X)$:

$$(\forall i < |X|_m)(\exists Z)(MCVP_p(a_{pump(X,i)}, E_{pump(X,i)}, G_{pump(X,i)}, Z) \\ \wedge Z[r(n, |s|, P, pump(i, X))] < p(|pump(i, X)|)).$$

Let $T'$ and $T''$ be the parse tree of $X'$ and $X''$ respectively. From a parse tree $T_i$ of $pump(X, i)$ we can construct a parse tree $T_{i+1}$ of $pump(X, i+1)$ by replacing the parse tree of $T''$ by $T'$. Again we note that this operation is $AC^0$ computable. So we have

$$(\forall i)(PT(i, X) \rightarrow PT(i+1, X)).$$

It is easy to see that $PT(0, X)$. Recall that $PT(i, X)$ is equivalent to a $\Sigma^B_0$ formula $\varphi(i, X)$ in the extended language and $\widehat{\mathbf{V}_{\mathrm{LCFL}}}$ proves $\Sigma^B_0$ induction. So we conclude that $(\forall i)PT(i, X)$. Thus again by Lemma 33 we have

$$(\forall i < |X|_m)(\exists T)Parse(|s|, l, n, |s|, P, T, X).$$

which proves the theorem.

## 7 Concluding Remarks

It is an interesting problem to determine the lower bound on the provability of the pumping lemma for context-free languages. For our theories one might conjecture that $\mathbf{V}_{\mathrm{LDCFL}}$ does not prove the theorem $(\forall n)(\forall s)(\forall P)PL(n, s, P)$.

As we have pointed out in the proof of Theorem 32, we need several combinatorial tools which lie within the theory for NL in order to execute the proof of the pumping lemma. Also it is essential in the proof that context free languages are definable in the theory.

Based on these observations, we come to the problem of whether $\mathbf{V}_{\mathrm{LDCFL}} + PATH$ proves $(\forall n)(\forall s)(\forall P)PL(n, s, P)$. Note that if it is not the case then $\mathbf{V}_{\mathrm{LDCFL}} + PATH$ cannot define (nondeterministic) context-free languages which gives a strong evidence implying that both NL and LOGDCFL is strictly contained in LOGCFL.

## References


1 F. Bédard, F. Lemieux and P. McKenzie, Extensions to Barrington's M-program model. Theoretical Computer Science, 107(1), 1993, pp.31–61.

2 A.Borodin, S.A.Cook, P.W.Dymond, W.L. Ruzzo, and M.Tompa, Two Applications of Inductive Counting for Complementation Problems. SIAM J. Comput. 18, 1989, pp. 559-578

3 S.A.Cook and P.Nguyen, Logical Foundations of Proof Complexity. Cambridge University Press 2010.

4 G. Gottlob, N. Leone and F. Scarcello, The complexity of acyclic conjunctive queries. Journal of the ACM 48(3) 2001, pp. 431–498.

5 J. E. Hopcroft and J. D. Ullman, Introduction to automata theory, languages and computation. Addison-Wesley, 1979.

6 S. Kuroda, Generalized Quantifier and a Bounded Arithmetic theory for LOGCFL. Archive for Mathematical Logic, 46(5-6), 2007, pp489-516.

7 P. McKenzie, K. Reinhardt and V. Vinay, Circuits and Context-Free languages. Computing and Combinatorics. Lecture Notes in Computer Science, 1627, 1999, pp.194-203

8 W.L. Ruzzo, Tree-size bounded alternation. Journal of Computer and System Sciences, 21(2), 1980 pp.139–154.

9 I. Sudborough, On the tape complexity of deterministic context-free language, Journal of ACM, 25(3), 1978, pp.405–414.

10 H. Venkateswaran, Properties that characterize LOGCFL. Journal of Computer and System Sciences, 43(2), 1991, pp.380–404.