Two–Sorted Theories for L, SL, NL and P

Electronic Colloquium on Computational Complexity, Report No. 17 (2005)

Phuong Nguyen
University of Toronto, April 2004. Revised Jan, 2005.

Abstract

We introduce "minimal" two–sorted first–order theories **VL**, **VSL**, **VNL** and **VP** that characterize the classes **L**, **SL**, **NL** and **P** in the same way that Buss's $\mathbf{S}_2^i$ hierarchy characterizes the polynomial time hierarchy. Our theories arise from natural combinatorial problems, namely the *st*–Connectivity Problem and the Circuit Value Problem. It turns out that **VL** is the same as Zambella's $\mathbf{\Sigma}_0^B$**-Rec**, **VP** is the same as Cook's $\mathbf{TV}^0$, and **VNL** and **VSL** are respectively the same as $\mathbf{V}^1$**-KROM** [8] and Kolokolova's $\mathbf{V}^1$**-SymKROM** [12]. Except for $\mathbf{VL} = \mathbf{\Sigma}_0^B$**-Rec**, establishing these equivalences is non-trivial.

1 Introduction

We study the logical characterization of the complexity classes $\mathbf{L}, \mathbf{SL}, \mathbf{NL}$ and $\mathbf{P}$. The first three are the classes of languages computable by respectively: *deterministic*, *symmetric nondeterministic*, and *nondeterministic* Turing machines using logarithmic space. **P** is the class of languages computable by deterministic Turing machines in polynomial time, or equivalently by *alternating* Turing machines using logarithmic space.

Each complexity class **C** is associated with a *function class* **FC**, i.e., the class of functions which grow at most polynomially in length, and whose graphs are in **C**. Thus **C** can be characterized by a theory $\mathcal{T}$ in the sense that the provably total functions in $\mathcal{T}$ are precisely the functions in **FC**. For example there have been a number of theories that characterize **P**: $\mathbf{S}_2^1$ [2], $\mathbf{V}^1$**-HORN** [7], $\mathbf{V}^1$ [4], $\mathbf{TV}^0$ [5] and **PV** [6] **VPV** [5]. The theory $\mathbf{S}_2^1$ is the first level in the hierarchy $\mathbf{S}_2^1 \subseteq \mathbf{S}_2^2 \ldots$ of single–sorted first–order theories which characterize the polynomial time hierarchy. The theory $\mathbf{V}^1$ is the two-sorted version of $\mathbf{S}_2^1$, while $\mathbf{V}^1$**-HORN** is defined using the syntactic class of Horn formulas, based on the fact that the Satisfiability Problem for this class of formulas is complete for **P**, and $\mathbf{TV}^0$ (which has been shown equivalent to $\mathbf{V}^1$**-HORN**) is a finitely axiomatizable theory which can be seen as *minimal* for polynomial-time reasoning. (The two theories $\mathbf{V}^1$ and $\mathbf{TV}^0$ lie in the hierarchy of two–sorted theories $\mathbf{V}^0 \subsetneq \mathbf{TV}^0 \subseteq \mathbf{V}^1 \subseteq \mathbf{TV}^1 \subseteq \ldots$ which we will not discuss here.)

In case of **NL**, the second-order theory $\mathbf{S}^{NLog}$ [3] is defined by formulating computation of logspace Turing machines. The theory $\mathbf{V}^1$**-KROM** [8], on the other hand, is similar to $\mathbf{V}^1$**-HORN** but is developed based on the fact that the satisfiability problem for propositional 2CNF formulas (conjunctive normal form formulas with 2 literals per clause, or the Krom formulas [9]) is complete for **NL**. Thus for $\mathbf{V}^1$**-KROM**, the Horn formulas are replaced by the Krom formulas.

Similar to $\mathbf{V}^1$**-KROM**, the theory $\mathbf{V}^1$**-SymKROM** has the same style as $\mathbf{V}^1$**-HORN**. It is defined in the same way as $\mathbf{V}^1$**-HORN**, using the class of the so-called *Symmetric Krom* formulas instead of the Horn formulas. It is based on the fact that the Satisfiability Problem for the Symmetric Krom formulas is complete for **SL**. The theories $\mathbf{V}^1$**-HORN**, $\mathbf{V}^1$**-KROM** and $\mathbf{V}^1$**-SymKROM** are developed based on the characterization of the corresponding classes in the context of Descriptive Complexity [9].

The theories we obtain are finitely axiomatizable and "minimal" for the corresponding classes. For **P** we will show that our theory **VP** is equivalent to $\mathbf{TV}^0$, and thus exhibits a finite axiomatization of $\mathbf{TV}^0$. For **L**, the theory **VL** for **L** obtained by our method, not surprisingly, is equivalent to Zambella's theory $\mathbf{\Sigma}_0^B$**-Rec** [14]. Consequently, it demonstrates the finite axiomatizability of $\mathbf{\Sigma}_0^B$**-Rec**. For **NL**, our theory **VNL** seems more natural than $\mathbf{V}^1$**-KROM** in the sense that it does not require the artificial syntactic requirement on the axioms. In this revision, we note that it has been shown [12] that $\mathbf{V}^1$**-KROM** and **VNL** are the same. With similar arguments one can see that $\mathbf{V}^1$**-SymKROM** $=$ **VSL**.

ISSN 1433-8092

Our theories grow out from the detailed treatment of two–sorted first–order logic in [4]. To prove the characterization of the classes by our theories, we follow the method used in [5] and introduce their conservative extensions which are universal and also "minimal". Indeed, we will discuss the universal, conservative extension $\overline{\mathbf{VNL}}$ of **VNL** in detail. From the fact that $\overline{\mathbf{VNL}}$ is a universal theory and a conservative extension of **VNL**, by applying the Herbrand's Theorem for $\overline{\mathbf{VNL}}$ we obtain the *Witnessing Theorem* for **VNL**: Any $\mathbf{\Sigma}_1$ theorem of **VNL** can be witnessed by an **NL** function.

It is intuitively true that for each of the classes mentioned above, we can develop a theory that characterizes the class by adding to the "base" theory $\mathbf{V}^0$ [4] an appropriate axiom corresponding to a complete problem of the class. However, proving this characterization might not be an easy task, depending on the particular class and the choice of the complete problem. Here, we choose the *Graph Connectivity Problem*, whose formalization we find more convenient than other combinatorial problems. Also, under various settings this problem becomes complete for the classes above in a very natural way: Computations of logspace Turing machines (which can be non-deterministic, *symmetric*, etc.) can be modeled by polynomial–size graphs (which will be directed, undirected, etc.) whose edge relations are precisely the "next configuration" relations of the machines. Thus, proving that these theories capture the corresponding classes may require minimal amount of work.

Note that the theories $\mathbf{S}^{Log}$ and $\mathbf{S}^{NLog}$ [3] also characterize **L** and **NL**, respectively. However these are second–order theories, while our theories are two–sorted first–order. The subtle difference is that in our setting, the binary string inputs are interpreted as the second sort objects, while in second-order theory the inputs are interpreted by the first-order objects (i.e., numbers).

1.1 Outline of the Paper

We introduce the theories in Section 2 and state the Main Theorem (2.9): the theories that we introduce characterize the corresponding classes in the same way that the $\mathbf{S}_2^i$ characterizes the polynomial time hierarchy. In Section 3 we prove the Main Theorem for the class **NL**. The same arguments can be applied to prove the Theorem for other classes. Then in Section 4 we show that **VP** is equivalent to Cook's theory $\mathbf{TV}^0$. Section 5 and 6 conclude the paper.

2 The Theories

2.1 The Graph Connectivity Problems

The classes mentioned in the introduction share a common source for a complete problem: the *Graph Connectivity Problem* (also known as the Reachability Problem). The problem is to decide, given a graph G and its two specific vertices s (for "source") and t (for "target"), whether there is a path from s to t. Thus when G is a directed graph, we have **STCONN**, a complete problem for **NL**; when G is directed and the outgoing degrees of the vertices of G are at most 1, we have the so-called **1-STCONN**, a complete problem for **L**; and when G is undirected, we have **USTCONN**, a complete problem for **SL**. For **P** we will consider **CVP**, the *Circuit Value Problem.* Note that there is another interpretation of this problem, i.e., the connectivity problem in a rather complicated kind of graphs (called *alternating graphs* in [11, p55]). We will not define these problems here, the readers may find them in many standard textbooks on complexity theory.

In order to show that **STCONN** is hard for **NL**, we construct, for each nondeterministic logspace Turing machine M and an input x, a graph G which represents the computation tree of M on x. More precisely, each vertex of G is labeled with a configuration of M on x, and there is a directed edge (u, v) in G iff v is labeled with a next possible configuration of the label of u. Here, a configuration of M consists of the state of M, the content of its work-tapes (but not the input), and the corresponding tape heads (including the input tape head). The configuration is encoded in the standard way, and since M works in logarithmic space, the graph G has polynomially many vertices. Note that this construction is possible in $\mathbf{AC}^0$. Similar arguments show that **USTCONN** and **1-STCONN** are hard for **SL** and **L**, respectively.

In developing the theories, we will use the following straightforward *polytime* algorithm that solve these problems. The idea is to evaluate, for each length k, all vertices that are reachable from the source s by a path of length *at most* k. Let *level* k denote this set. Then level 0 consists of only s, and level a contains t iff there is a path from s to t, where a is the number of vertices in G. Also, it is straightforward to compute level $k+1$ inductively from level k using the edge relation in G. The algorithm is presented in Figure 1 below. Here V_G and E_G are respectively the set of vertices and edges in G. The vertices on level k are recorded on the row $Z^{[k]}$ of a 2–dimensional array Z. Note that E_G is the set of ordered pairs.

Input: $G = (V_G, E_G)$ and $s, t \in V_G$, $|V_G| = a$.

1. $Z := \emptyset$ (Z is the 2–dimensional array, the row $Z^{[k]}$ keeps vertices on level k.)
2. $Z^{[0]} := \{s\}$
3. For $k = 0 \mathrel{..} (a-1)$ do
   - $Z^{[k+1]} := Z^{[k]}$
   - For $u \in Z^k$, $v \in V_G$, if $(u, v) \in E_G$ then $Z^{[k+1]} := Z^{[k+1]} \cup \{v\}$
4. If $t \in Z^{[a]}$ then output YES; otherwise output NO.

Figure 1: A polynomial time algorithm for Connectivity

2.2 VL, VSL and VNL

We refer to [4] and [5] for the syntax and semantics of the two–sorted first–order logic that we are using. Note that there are two sorts of variables: the *number variables*, denoted by $x, y, z, \ldots$; and the *string variables*, denoted by $X, Y, Z, \ldots$. The number variables are intended to range over the set $\mathbb{N}$ of natural numbers, which are interpreted as unary strings. The string variables are intended to range over the set of finite subsets of $\mathbb{N}$, which are represented by finite binary strings.

The theory $\mathbf{V}^0$ serves as the "base" theory for all of our theories. It is axiomatized by (i) the set of defining axioms for the symbols of the underlying vocabulary

$$\mathcal{L}_A^2 = [0, 1, +, \cdot, |\;|; =_1, =_2, \leq, \in]$$

and (ii) the *comprehension axiom scheme over* $\mathbf{\Sigma}_0^B$ *formulas*, $\mathbf{\Sigma}_0^B$-**COMP**. Here $\mathbf{\Sigma}_0^B$ is the set of all formulas whose quantifiers are bounded and are over numbers only. Then $\mathbf{\Sigma}_0^B$-**COMP** is the set of all formulas of the form

$$\exists X \leq y \forall z < y (X(z) \leftrightarrow \phi(z))$$

for ϕ a $\mathbf{\Sigma}_0^B$ formula, and X does not occur free in ϕ.

Other important sets of formulas include $\mathbf{\Sigma}_1^B$ and $\mathbf{\Pi}_1^B$, the sets of formulas of the form $\exists \vec{X} \leq \vec{t} \varphi(\vec{X})$ and $\forall \vec{X} \leq \vec{t} \varphi(\vec{X})$ respectively, where $\varphi(\vec{X})$ is a $\mathbf{\Sigma}_0^B$ formula.

We often use $=$ for both $=_1$ and $+_2$, the meaning will be clear from the context. Also, we will denote the membership relation $x \in Z$ simply by $Z(x)$. We will use $\langle x, y \rangle$ for the pairing function:

$$\langle x, y \rangle = (x+y)(x+y+1) + x$$

We will simply use $Z(x,y)$ for $Z(\langle x,y\rangle)$. Using the pairing function, we can encode a 2–dimensional Boolean array in a string variable as follows. The row k of an array Z, $Z^{[k]}$ is defined as

$$|Z^{[k]}| \leq |Z| \wedge Z^{[k]}(x) \leftrightarrow Z(k,x)$$

Each of our theories is axiomatized by $\mathbf{V}^0$ together with an appropriate axiom, the main part of which encodes the algorithm described in Figure 1 in the following way. Suppose that there are a vertices in G, i.e., $|V_G| = a$, then V_G can be represented simply by $\{0, \ldots, a-1\}$ (and it suffices therefore to mention only a). Also, the edge relation E_G is given by the string variable E: $E(i,j)$ holds iff $(i,j) \in E_G$. Then the levels k, for $0 \leq k \leq n$, are stored in a string variable Z which we view as coding a 2–dimensional array: The row k of Z consists precisely of those numbers j which are on the level k computed by the algorithm in Figure 1.

Without loss of generality, we identify the source s with 0, and the target t with 1. The following formula encodes the polytime algorithm given in Figure 1. (LC stands for Logspace Computation.)

Definition 2.1 (LC)

$$\begin{aligned} LC(a,E,Z) \equiv & |Z| \leq \langle a,a\rangle \wedge Z(0,0) \wedge \forall i < a,\ 0 < i \supset \neg Z(0,i) \wedge \\ & \forall k,i < a,\ Z(k+1,i) \leftrightarrow [Z(k,i) \vee \exists j < a,\ E(j,i) \wedge Z(k,j)]. \end{aligned} \tag{1}$$

It is easy to check that $Z(k,i)$ holds iff there is a path from 0 to i of length at most k. In other words, Z is the string which "calculates" all vertices reachable from 0 in the graph E.

Note that given a and E, it is true that there exists Z that satisfies $LC(a,E,Z)$. However this is not provable in the theory $\mathbf{V}^0$. Nevertheless, $\mathbf{V}^0$ proves that if such Z exists, it is unique.

Lemma 2.2 $\mathbf{V}^0 \vdash (LC(z,E,Z_1) \wedge LC(z,E,Z_1)) \supset Z_1 = Z_2$

Proof The Lemma follows easily from the fact that $\mathbf{V}^0$ proves the *number induction scheme* on $\mathbf{\Sigma}_0^B$ formulas, i.e., the set of all formulas of the form

$$\varphi(0) \wedge \forall x < y(\varphi(x) \supset \varphi(y)) \supset \varphi(y)$$

where φ is a $\mathbf{\Sigma}_0^B$ formula. □

Now, the theories $\mathbf{VL}, \mathbf{VSL}, \mathbf{VNL}$ simply state the existence of Z given particular conditions on E:

Definition 2.3 ($\mathbf{VL}, \mathbf{VSL}, \mathbf{VNL}$)

$$\mathbf{VL} =_{\text{defn}} \mathbf{V}^0 + \forall i < a \exists! j < a\ E(i,j) \supset \exists Z\ LC(a,E,Z), \tag{2}$$

$$\mathbf{VSL} =_{\text{defn}} \mathbf{V}^0 + [\forall i,j < a\ E(i,j) \leftrightarrow E(j,i)] \supset \exists Z\ LC(a,E,Z), \tag{3}$$

$$\mathbf{VNL} =_{\text{defn}} \mathbf{V}^0 + \exists Z\ LC(a,E,Z). \tag{4}$$

Note that the length of Z is bounded by $\langle a,a\rangle$. Therefore the theories are bounded. Also, $\mathbf{V}^0$ is finitely axiomatizable [7]. Therefore $\mathbf{VL}$, $\mathbf{VSL}$ and $\mathbf{VNL}$ are all finitely axiomatizable.

Corollary 2.4 *The theories* $\mathbf{VL}, \mathbf{VSL}$ *and* $\mathbf{VNL}$ *are finitely axiomatizable.*

The theory $\mathbf{\Sigma}_0^B$-**Rec** introduced by Zambella [14] is $\mathbf{V}^0$ together with the following recursion scheme, which asserts the existence of a path in a directed graph whose vertices have outgoing degrees at least 1. Formally this can be taken as the set of all formulas of the form

$$\forall i < a \exists j < a \varphi(i,j) \supset \exists Z, \mathit{fval}(0,Z) = 0 \wedge \forall w < a\, \varphi(\mathit{fval}(w,Z), \mathit{fval}(w+1,Z)). \tag{5}$$

where φ is a $\mathbf{\Sigma}_0^B$ formula not involving Z.

Here $fval(w, Z)$ is the function that extracts the value at w of a function coded by Z. The coding scheme should be simple, i.e., $fval(x, Z)$ can be defined by an $\mathbf{AC}^0$ formula. Here we will assume the following coding scheme:

$$fval(x, Z) = z \leftrightarrow Z(x, z) \wedge \forall y < |Z|,\ Z(x, y) \supset y = z.$$

It is not surprising that **VL** is the equivalent to $\mathbf{\Sigma}_0^B$-**Rec**, and thus **VL** exhibits a finite axiomatization of $\mathbf{\Sigma}_0^B$-**Rec**.

Lemma 2.5 **VL** $=$ $\mathbf{\Sigma}_0^B$-**Rec**.

Corollary 2.6 $\mathbf{\Sigma}_0^B$-**Rec** *is finitely axiomatizable.*

Proof of Lemma 2.5
a) **VL** $\subseteq$ $\mathbf{\Sigma}_0^B$-**Rec**: Suppose that E is a string such that $\forall i < a\ \exists! j < aE(i, j)$. We need to prove in $\mathbf{\Sigma}_0^B$-**Rec** that there exists Z such that $LC(a, E, Z)$ holds.

Let $\varphi(i, j) \equiv E(i, j)$, then the precondition of (5) is satisfied. Let Z' be the string whose existence is guaranteed as in (5). Then $fval(w, Z')$ is the only vertex that is reachable from the source 0 by a path of length exactly w.

Now in the desired Z that satisfies $LC(a, E, Z)$, level i consists of all vertices that are reachable from the source by path of length at most i. Thus Z is defined from Z' using $\mathbf{\Sigma}_0^B$-**COMP** as follows.

$$Z(i, j) \leftrightarrow \exists w \le i,\ j = fval(w, Z').$$

b) $\mathbf{\Sigma}_0^B$-**Rec** $\subseteq$ **VL**: Let $\varphi(i, j)$ be a $\mathbf{\Sigma}_0^B$ formula such that $\forall i < a \exists j < a\varphi(i, j)$. We need to prove in **VL** the existence of Z that satisfies (5). Note that for each i, there may be more than one j such that $\varphi(i, j)$ holds. Hence we will take the the smallest such j. Formally, using $\mathbf{\Sigma}_0^B$-**COMP**, $\mathbf{V}^0$ proves that

$$\exists E < \langle a, a\rangle,\ E(i, j) \leftrightarrow i < a \wedge j < a \wedge \varphi(i, j) \wedge \forall \ell < j\ \neg\varphi(i, \ell).$$

Let Z' be the string that satisfies $LC(a, E, Z')$. Then a vertex j is reachable from the source 0 by a path of length exactly k iff it is in $Z'^{[k]}$ but not in any $Z'^{[k']}$ where $k' < k$. Thus the string Z of (5) is defined as follows:

$$fval(w, Z) = z \leftrightarrow Z'(w, z) \wedge \forall w' < w\ \neg Z(w', z).$$

Hence in $\mathbf{V}^0$, Z can be defined using $\mathbf{\Sigma}_0^B$-**COMP** by:

$$|Z| \le \langle a, a\rangle \wedge\ [\forall w, z < a,\ Z(w, z) \leftrightarrow Z'(w, z) \wedge \forall w' < w\ \neg Z(w', z)]$$

□

2.3 VP

Now we will define the theory **VP** for **P** using **CVP**, the *Circuit Value Problem.* This is the problem of deciding, given a Boolean circuit and its input, whether the output of the circuit is 1. Here we restrict to monotone circuits with unbounded fan-ins, but the problem remains equally hard.

The following polytime algorithm computes the output of a Boolean circuit G which has a gates. The gates of G are labeled with $0, \ldots, (a-1)$, and the edges (i.e., wires) of G are given by E_G: $(u, v) \in E_G$ iff the output of the gate labeled with v is connected to the input of the gate labeled with u.

The algorithm runs in a loops. The idea is to identify all the gates whose values are 1. In loop 0 we simply single out the input gates with the value 1. In each subsequent loop $k+1$ we identify "as many more gates as possible": all gates that have been identified in loop k, together with all

- $\vee$–gates that have at least one input from the gates we have in loop k;
- $\wedge$–gates that have all inputs from the gates we have in loop k.

(Note that we consider only monotone circuits.) The gates that we identify in loop k are stored in the row $Z^{[k]}$ of an array Z. Also, assume that the output gate is labeled with 0.

Input: Circuit G with a gates labeled with $0, \ldots, (a-1)$, output gate is labeled with 0.

1. $Z := \emptyset$
2. $Z^{[0]} := \{\text{input gates with value } 1\}$
3. For $k = 0 \,..\, (a-1)$ do
   - $Z^{[k+1]} := Z^{[k]}$
   - For $0 \leq u < a$
     - if u is an $\wedge$–gate, and for all v such that $(u,v) \in E_G$, $v \in Z^{[k]}$ then $Z^{[k+1]} := Z^{[k+1]} \cup \{v\}$
     - if u is a $\vee$–gate, and there is a v such that $(u,v) \in E_G$ and $v \in Z^{[k]}$ then $Z^{[k+1]} := Z^{[k+1]} \cup \{v\}$
4. If $0 \in Z^{[a]}$ then output YES; otherwise output NO.

Figure 2: A polynomial time algorithm for **CVP**

Now we define ALC. Here E encodes the edge (i.e., wires) in a Boolean circuit where the gates have unbounded fan-ins: $E(i,j)$ holds iff the output of the node labeled j is connected to an input of the gate labeled by i. We also assume that the $\wedge$–gates are labeled by the even numbers, and the $\vee$–gates are labeled by the odd numbers.

We can assume further that there is only one 1–bit input, and it is given at the input gate labeled with 1. In the formula ALC below the 2–dimensional array Z is used to evaluate all gates of the circuit: $Z(k,i)$ holds iff the gate labeled i is identified in the loop k of the algorithm. (Thus we deliberately give it the same name with the array in the algorithm.)

Definition 2.7 (ALC)

$$\begin{aligned} ALC(a,E,Z) \equiv & |Z| \leq \langle a,a \rangle \wedge Z(0,1) \wedge \forall i < a\ i \neq 1 \supset \neg Z(0,i) \wedge \\ & \forall i,k < a,\ \ Z(k+1,2i) \leftrightarrow [Z(k,2i) \vee \forall j < a\ E(2i,j) \supset Z(k,j)] \wedge \\ & \qquad Z(k+1,2i+1) \leftrightarrow [Z(k,2i+1) \vee \exists j < a\ E(2i+1,j) \wedge Z(k,j)] \end{aligned} \tag{6}$$

(Note that although $Z(k,i)$ are also defined for $a \leq i < 2a$, the values $Z(k,i)$ where $i \geq a$ are irrelevant.)

Definition 2.8 (VP) $\mathbf{VP} =_{\text{defn}} \mathbf{V}^0 + \exists Z ALC(a,E,Z)$.

Since Z has length bounded by $\langle a,a \rangle$, **VP** is a bounded theory. Also, since $\mathbf{V}^0$ is finitely axiomatizable [7], so is **VP**.

Note that **P** is the same as the class of languages computable by logspace *alternating Turing machine*. Furthermore, evaluating the acceptance of a computation of an alternating Turing machine can be roughly viewed as computing the output of a Boolean circuit. (For logspace ATMs, the circuit will be of polynomial size.) The formula ALC encodes the above polytime algorithm for **CVP** in the same way that LC encodes the polytime algorithm for **STCONN** given in Figure 1.

2.4 The Main Theorem

Recall that $\mathbf{\Sigma}_1^B$ and $\mathbf{\Pi}_1^B$ formulas are of the form $\exists \vec{X} \le \vec{t}\varphi(\vec{X})$ and $\forall \vec{X} \le \vec{t}\varphi(\vec{X})$, respectively, where φ is a $\mathbf{\Sigma}_0^B$ formula. We say that a relation $R(\vec{x},\vec{X})$ is $\mathbf{\Delta}_1^B$–definable in a theory $\mathcal{T}$ if there are a $\mathbf{\Sigma}_1^B$ formula $\varphi(\vec{x},\vec{X})$ and a $\mathbf{\Pi}_1^B$ formula $\psi(\vec{x},\vec{X})$ such that

$$R(\vec{x},\vec{X}) \text{ iff } \varphi(\vec{x},\vec{X}), \qquad \text{and} \qquad \mathcal{T} \vdash \varphi(\vec{x},\vec{X}) \leftrightarrow \psi(\vec{x},\vec{X})$$

Also note that by Parikh's Theorem, for each of our theories the class of $\mathbf{\Sigma}_1$–definable functions is the same as the class of $\mathbf{\Sigma}_1^B$–definable functions.

Theorem 2.9 (Main Theorem) *For each class* **C** *of the classes* **L**, **SL**, **NL** *and* **P**, *the functions in* **FC** *are precisely the* $\mathbf{\Sigma}_1$*–definable functions of* **VC**, *and the relations in* **C** *are precisely the* $\mathbf{\Delta}_1^B$*–definable relations of* **VC**.

In the next Section we will prove this Theorem for the case of **NL** (Corollary 3.12). Similar arguments can be applied to other classes.

3 Characterizing NL by VNL

3.1 Defining NL Relations and Functions in VNL

Theorem 3.1 *The* **NL** *relations are* Δ_1^B *definable in* **VNL**.

Proof Let $R(\vec{x},\vec{X})$ be an **NL** relation, which is decided by a non-deterministic logspace Turing machine M. We can assume that M has only one accepting configuration (e.g, upon entering the accepting state, it erases all of its work-tapes content). Then, there is an $\mathbf{AC}^0$ relation (or equivalently, a $\mathbf{\Sigma}_0^B$ formula) $STEP_{\mathsf{M}}(\vec{x},\vec{X},i,j)$ such that $STEP_{\mathsf{M}}(\vec{x},\vec{X},i,j)$ holds if and only if j codes a next configuration of M (on inputs $(\vec{x},\vec{X})$) of the configuration coded by i. Note that $STEP_{\mathsf{M}}$ depends on the encoding of M's configurations. We do not go into the details of such encoding, but we assume that 0 always codes the starting configuration, and 1 always codes the only accepting configuration. Furthermore, there is an $\mathcal{L}_A^2$ number term $nconfs_{\mathsf{M}}(\vec{x},\vec{n})$ that bounds the number of different configurations of M on input $(\vec{x},\vec{X})$ where $|\vec{X}| = \vec{n}$ (we write t_{M} for $nconfs_{\mathsf{M}}(\vec{x},|\vec{X}|)$).

Definition 3.2 *For each non-deterministic logspace Turing machine* M, $GRAPH_{\mathsf{M}}(\vec{x},\vec{X},E)$ *is the following* $\mathbf{\Sigma}_0^B$ *formula:*

$$GRAPH_{\mathsf{M}}(\vec{x},\vec{X},E) \equiv |E| \le \langle t_{\mathsf{M}}, t_{\mathsf{M}}\rangle \wedge \forall i,j < t_{\mathsf{M}}\ E(i,j) \leftrightarrow STEP_{\mathsf{M}}(\vec{x},\vec{X},i,j). \tag{7}$$

And $Graph_{\mathsf{M}}(\vec{x},\vec{X})$ *is the* $\mathbf{AC}^0$ *function whose graph is* $GRAPH_{\mathsf{M}}$*:*

$$Graph_{\mathsf{M}}(\vec{x},\vec{X})(i,j) \leftrightarrow i < t_{\mathsf{M}} \wedge j < t_{\mathsf{M}} \wedge STEP_{\mathsf{M}}(\vec{x},\vec{X},i,j) \tag{8}$$

Then, $R(\vec{x}, |\vec{X}|)$ is represented by the following $\mathbf{\Sigma}_1^B$ and $\mathbf{\Pi}_1^B$ formulas

$$
\begin{aligned}
R(\vec{x}, |\vec{X}|) &\Leftrightarrow \exists E, Z,\ \mathit{GRAPH}_{\mathsf{M}}(\vec{x}, \vec{X}, E) \wedge LC(t, E, Z) \wedge Z(t, 1) && (9)\\
&\Leftrightarrow \forall E, Z,\ (\mathit{GRAPH}_{\mathsf{M}}(\vec{x}, \vec{X}, E) \wedge LC(t, E, Z)) \supset Z(t, 1) && (10)
\end{aligned}
$$

(Note that the lengths of E, Z are bounded, see (1) and (7).)

It remains to show that the two formulas in (9) and (10) are equivalent in **VNL**.

(9) $\Rightarrow$ (10) E is unique because $E = \mathit{Graph}_{\mathsf{M}}(\vec{x}, \vec{X})$. The uniqueness of Z follows from the uniqueness of E and Lemma 2.2.

(10) $\Rightarrow$ (9) E exists because $E = \mathit{Graph}_{\mathsf{M}}(\vec{x}, \vec{X})$. Z exists using LC (1). $\square$

Theorem 3.3 *The* **NL** *functions are* $\mathbf{\Sigma}_1^B$ *definable in* **VNL**.

Before proving this theorem, we prove the following lemma, which states that **VNL** proves the existence of "combined" computations and "combined" evaluations, i.e., the existence of multiple E and multiple Z as in (9).

Lemma 3.4 *For each non-deterministic logspace Turing machine* M *that works on input* $(z, \vec{x}, \vec{X})$,

$$\mathbf{VNL} \vdash \exists! E_1, Z_1\ \forall z < a,\ [\mathit{GRAPH}_{\mathsf{M}}(z, \vec{x}, \vec{X}, E_1^{[z]}) \wedge LC(a, E_1^{[z]}, Z_1^{[z]})]. \qquad (11)$$

In other words, for each value of z, $E_1^{[z]}$ is the graph of computations of M on input $(z, \vec{x}, \vec{X})$, and $Z_1^{[z]}$ is the reachability relation of $E_1^{[z]}$: $Z_1^{[z]}(k, i)$ holds if and only if there is a path of length $\leq k$ from 0 to i in the graph $E_1^{[z]}$.

Proof of Lemma 3.4 The uniqueness of E_1 and Z_1 can be proved in similar way as in Lemma 2.2. We will prove their existence. Indeed, $E_1^{[z]}$ is just $\mathit{Graph}_{\mathsf{M}}(z, \vec{x}, \vec{X})$, therefore E_1 can be defined in $\mathbf{V}^0$ using $\mathbf{\Sigma}_0^B$-**COMP**, i.e.,

$$E_1(z, i, j) \leftrightarrow z < a \wedge i < a \wedge j < a \wedge \mathit{Graph}(z, \vec{x}, \vec{X})(\langle i, j \rangle)$$

Now we prove the existence of Z_1. Note that for each z, by definition (4), a string Z exists which calculates the vertices reachable from 0 in the graph $E_1^{[z]}$. Hence we need to show the existence of all of them in the "combined" string Z_1. For this, we create a "big graph" E' which has a copy of each "small graph" $E_1^{[z]}$, for all $0 \leq z < a$. We also connect the source of each of these copies with a single source in E'. Then in general, there is a path of length k in $E_1^{[z]}$ from the source to v iff there is a path of length $k+1$ from the source (of E') to the copy of v in E'. (This is true except for $E_1^{[0]}$: there is a path of length k in $E_1^{[0]}$ from the source to v iff there is a path of length k from the source (of E') to the copy of v in E'.)

The "big graph" E' is obtained by concatenating the rows of E_1. The edge (i, j) is in $E_1^{[z]}$ iff the edge $(az + i, az + j)$ is in E'. Also, $(0, az) \in E'$, for $0 \leq z < a$. Formally E' is defined as

$$[\forall z, i, j < a,\ E'(az + i, az + j) \leftrightarrow E_1^{[z]}(i, j)] \ \wedge\ [\forall z < a,\ E'(0, az)].$$

Note that $|E'| < \langle a^2, a^2 \rangle$. Let Z' be the string existed that satisfies $LC(a^2, E', Z')$. Then, for $k \leq a, i < a$:

$$Z_1^{[0]}(k, i) \leftrightarrow Z'(k, i) \quad \text{and} \quad Z_1^{[z]}(k, i) \leftrightarrow Z'(k + 1, az + i),\ \text{for } 1 \leq z < a.$$

Since **VNL** proves the existence of Z', it also proves the existence of Z_1. $\square$

Proof of Theorem 3.3 We consider the cases of string functions and number functions separately.

First, suppose that $F(\vec{x}, \vec{X})$ is an **NL** string function, i.e., F is p-bounded by an $\mathcal{L}_A^2$ term s, and there is a non-deterministic logspace Turing machine M that computes the bit graph $R(z, \vec{x}, \vec{X})$ of F. Let E_1, Z_1 be the strings which exist by Lemma 3.4, where $a = max\{s, nconfs_{\mathsf{M}}(\vec{x}, |\vec{X}|)\}$. Then for each "bit z", $z \in F(\vec{x}, \vec{X})$ iff M accepts the input $(z, \vec{x}, \vec{X})$, i.e., in the "small graph" $E_1^{[z]}$ there is a path from the source 0 to the target 1. In other words, F can be defined as follows

$$F(\vec{x}, \vec{X}) = Y \leftrightarrow \forall z < s\ [Y(z) \leftrightarrow Z_1^{[z]}(a, 1)].$$

It follows that **VNL** can $\mathbf{\Sigma}_1^B$ define F.

Now, suppose that $f(\vec{x}, \vec{X})$ is an **NL** number function. Suppose that $f(\vec{x}, \vec{X}) < s(\vec{x}, |\vec{X}|)$, and the graph $R(y, \vec{x}, \vec{X})$ of f is computable by a non-deterministic logspace Turing machine M. Let E_1, Z_1 be the strings as in (11) (where $a = max\{s(\vec{x}, |\vec{X}|), nconfs_{\mathsf{M}}(\vec{x}, |\vec{X}|)\}$). The f is defined by the formula

$$y = f(\vec{x}, \vec{X}) \leftrightarrow Z_1^{[y]}(a, 1)$$

Thus f is $\mathbf{\Sigma}_1$–definable in **VNL**. $\square$

3.2 The Vocabulary $\mathcal{L}_{\mathbf{NL}}$ of NL Functions

Let F_{STCONN} be the (partial) function whose graph is LC, i.e.,

$$F_{STCONN}(x, E) = Z \leftrightarrow LC(x, E, Z). \tag{12}$$

Then, intuitively, F_{STCONN} is complete for **NL**. Consequently, the $\mathbf{AC}^0$ closure of F_{STCONN} represents precisely the **NL** functions. We will formally prove this claim. Define $\mathcal{L}_{\mathbf{NL}}$ as follows.

Definition 3.5 ($\mathcal{L}_{\mathbf{NL}}$) *$\mathcal{L}_{\mathbf{NL}}$ is the smallest vocabulary that contains $\mathcal{L}_{\mathbf{FAC}^0} \cup \{F_{STCONN}\}$ such that for each open formula $\alpha(z, \vec{x}, \vec{X})$ over $\mathcal{L}_{\mathbf{NL}}$ and term $t = t(\vec{x}, \vec{X})$ of $\mathcal{L}_A^2$, there is a string function $F_{\alpha,t}$ and a number function $f_{\alpha,t}$ of $\mathcal{L}_{\mathbf{NL}}$ with defining axiom(s)*

$$F_{\alpha,t}(\vec{x}, \vec{X})(z) \leftrightarrow z < t \wedge \alpha(z, \vec{x}, \vec{X}) \tag{13}$$

$$f_{\alpha,t}(\vec{x}, \vec{X}) \leq t(\vec{x}, \vec{X}) \tag{14}$$

$$f_{\alpha,t}(\vec{x}, \vec{X}) < t \supset \alpha(f_{\alpha,t}(\vec{x}, \vec{X}), \vec{x}, \vec{X})) \tag{15}$$

$$z < f_{\alpha,t}(\vec{x}, \vec{X}) \supset \neg\alpha(z, \vec{x}, \vec{X}) \tag{16}$$

Lemma 3.6 *A relation is in* **NL** *if and only if*

a) *it is represented by an open $\mathcal{L}_{\mathbf{NL}}$ formula;*
b) *it is represented by a $\mathbf{\Sigma}_0^B(\mathcal{L}_{\mathbf{NL}})$ formula.*

Proof Suppose that $R(\vec{x}, \vec{X})$ is an **NL** relation. Recall the definition of the $\mathbf{AC}^0$ function $Graph_{\mathsf{M}}$ in (8). It is evident from the proof of Theorem 3.1 that R is represented by the following open formula over $\mathcal{L}_{\mathbf{NL}}$:

$$R(\vec{x}, \vec{X}) \Leftrightarrow F_{STCONN}(t, Graph_{\mathsf{M}}(\vec{x}, \vec{X}))(t, 1),$$

where M is a non-deterministic Turing machine that computes R, and $t = nconfs_{\mathsf{M}}(\vec{x}, \vec{X})$ is the upper bound on the number of different of configurations of M.

It remains to show that any $\mathbf{\Sigma}_0^B(\mathcal{L}_{\mathbf{NL}})$ formula $\varphi(\vec{x}, \vec{X})$ can be evaluated by **NL** machines. This is proved by induction on the structure of φ. The base case is straightforward. For the induction step, we need to use the fact that **NL** is closed under (i) negation, (ii) Boolean operations, and (iii) bounded number quantifications. Note that (i) follows from the Immermann-Szelepcsényi result [10, 13], and (ii) and (iii) are straightforward. □

Corollary 3.7 *The functions in $\mathcal{L}_{\mathbf{NL}}$ represent precisely the* **NL** *functions.*

Proof Let $F(\vec{x}, \vec{X})$ be an **NL** string function. Then by definition,

$$F(\vec{x}, \vec{X})(i) \Leftrightarrow i < p \wedge R(i, \vec{x}, \vec{X})$$

for some polynomial p and some **NL** relation R. By Lemma 3.6, there is an open $\mathcal{L}_{\mathbf{NL}}$ formula φ that represents R. Consequently, F is the function of $\mathcal{L}_{\mathbf{NL}}$ defined by (13).

The case of **NL** number function is similar. □

3.3 The Open Theory $\overline{\mathbf{VNL}}$ and The Witnessing Theorem for VNL

Note that LC (1) is not an open formula. Therefore, the defining axiom for F_{STCONN} (12) is not really an open formula. However, the existential quantifier in LC can be eliminated using the following technique from [5]. Let f_{LC} be the $\mathbf{AC}^0$ function defined by

$$f_{LC}(k, i, a, E, Z) = \min\{z : z < a, \text{ and } E(z, i) \wedge Z(k, z)\}.$$

(Thus $f_{LC}(k, i, a, E, Z) = f_{\alpha,a}$ where $\alpha \equiv E(z, i) \wedge Z(k, z)$; see (14), (15), (16).) Now we have

$$\exists j < a\ E(j, i) \wedge Z(k, j) \ \leftrightarrow\ f_{LC}(k, i, a, E, Z) < a$$

Therefore we obtain the following open formula (in $\mathcal{L}_A^2 \cup \{f_{LC}\}$) which is equivalent to LC:

$$\begin{aligned} LC'(a, E, Z) \equiv & Z(0, 0) \ \wedge\ [i < a \supset \neg Z(0, i)] \ \wedge \\ & k < a \wedge i < a \ \supset\ [Z(k + 1, i) \leftrightarrow (Z(k, i) \vee f_{LC}(k, i, a, E, Z) < a)]. \end{aligned} \tag{17}$$

Definition 3.8 ($\overline{\mathbf{VNL}}$) $\overline{\mathbf{VNL}}$ *is the theory over $\mathcal{L}_{\mathbf{NL}}$, where F_{STCONN} has the defining axiom using LC'* (17)*, and other functions are defined according to Definition 3.5.*

Lemma 3.9 *1. For each $\mathbf{\Sigma}_0^B(\mathcal{L}_{\mathbf{NL}})$ formula φ, there is an open $\mathcal{L}_{\mathbf{NL}}$–formula φ' such that*

$$\mathbf{VNL} \vdash \varphi' \leftrightarrow \varphi$$

*2. $\overline{\mathbf{VNL}} \vdash \mathbf{\Sigma}_0^B(\mathcal{L}_{\mathbf{NL}})$-**COMP**.*

Proof 1) is proved by structural induction on φ. The case of quantifier in the induction step is proved by the same method that we have used to eliminate the quantifier in LC.

2) follows from 1) and the fact that

$$Z = F_{\alpha,t} \leftrightarrow |Z| \leq t \wedge \forall z < t Z(z) \leftrightarrow \alpha(z)$$

(See (13).) □

Lemma 3.10 $\overline{\mathbf{VNL}}$ *is a conservative extension of* $\mathbf{VNL}$.

Proof To show that $\overline{\mathbf{VNL}}$ extends $\mathbf{VNL}$, we show that $\overline{\mathbf{VNL}}$ proves the existence of Z in (4). This is immediate, since $Z = F_{STCONN}(a, E)$.

It remains to show that $\overline{\mathbf{VNL}}$ is conservative over $\mathbf{VNL}$. It suffices to show that $\mathbf{VNL} \vdash \mathbf{\Sigma}_0^B(\mathcal{L}_{\mathbf{NL}})$-**COMP**. Note that the functions of $\mathcal{L}_{\mathbf{NL}}$ are $\mathbf{\Sigma}_1$–definable in $\mathbf{VNL}$ (Theorem 3.3). Consider a $\mathbf{\Sigma}_0^B(\mathcal{L}_{\mathbf{NL}})$ formula $\varphi(z)$. By Lemma 3.6 it represents an $\mathbf{NL}$ relation, i.e., the bit graph of an $\mathbf{FNL}$ function F. By Theorem 3.3, F is $\mathbf{\Sigma}_1^B$–definable in $\mathbf{VNL}$, thus $\mathbf{VNL}$ proves the comprehension axiom for $\varphi(z)$. □

Theorem 3.11 (Witnessing Theorem for VNL) $\mathbf{\Sigma}_1^B$ *theorems of* $\mathbf{VNL}$ *are witnessed by* $\mathbf{NL}$ *functions.*

Proof Suppose that $\exists Y \varphi(\vec{x}, \vec{X}, Y)$ is a $\mathbf{\Sigma}_1^B$ theorem of $\mathbf{VNL}$. Then by Lemma 3.10, it is also a theorem of $\overline{\mathbf{VNL}}$. Since $\overline{\mathbf{VNL}}$ is a universal theory, by the Herbrand Theorem, there is a function of $\mathcal{L}_{\mathbf{NL}}$ such that $\overline{\mathbf{VNL}} \vdash \varphi(\vec{x}, \vec{X}, F(\vec{x}, \vec{X}))$. □

Corollary 3.12 *A function is in* $\mathbf{FNL}$ *if and only if it is* $\mathbf{\Sigma}_1^B$*-definable in* $\mathbf{VNL}$. *A relation is in* $\mathbf{NL}$ *if and only if it is* $\mathbf{\Delta}_1^B$*-definable in* $\mathbf{VNL}$.

4 $\mathbf{VP} = \mathbf{TV}^0$

In [5] Cook introduces the theory $\mathbf{TV}^0$, and shows that it characterizes $\mathbf{P}$. He also shows that $\mathbf{TV}^0$ is equivalent to $\mathbf{V}^0 + \mathbf{\Sigma}_0^B$-**BIT-REC**, where $\mathbf{\Sigma}_0^B$-**BIT-REC**, the *bit-recursion* scheme for $\mathbf{\Sigma}_0^B$ formulas, is the following scheme:

$$\exists X \leq a \forall x < a,\ X(x) \leftrightarrow \varphi(x, X^{<x}), \tag{18}$$

where φ is $\mathbf{\Sigma}_0^B$, and $X^{<x}$ is the $\mathbf{AC}^0$ function defined by

$$X^{<x}(z) \leftrightarrow z < x \wedge X(z).$$

Here we will show that $\mathbf{TV}^0 = \mathbf{VP}$, and this verifies the claim in [5] that $\mathbf{TV}^0$ is finitely axiomatizable.

Theorem 4.1 $\mathbf{VP} = \mathbf{TV}^0$.

Corollary 4.2 ([5]) $\mathbf{TV}^0$ *is finitely axiomatizable.*

Proof of Theorem 4.1 Note that $\exists Z ALC(a, E, Z)$ (see Definition 2.8) is a special form of the $\mathbf{\Sigma}_0^B$-**BIT-REC** axioms. Hence $\mathbf{VP}$ is a sub-theory of $\mathbf{TV}^0$. It remains to show that the $\mathbf{\Sigma}_0^B$-**BIT-REC** axioms hold in $\mathbf{VP}$.

Let $\varphi(x, Z)$ be a $\mathbf{\Sigma}_0^B$ formula. We need to prove in $\mathbf{VP}$ the existence of X that satisfies (18). We will construct a monotone circuit encoded by a string variable E so that from the string Z which satisfies $ALC(a, E, Z)$ we can extract the required value of X. We will show how to construct such monotone circuit C; encoding of C by a string E is straightforward, and is omitted.

Note that (18) gives a recursive definition of the *initial segments* of X. The circuit C has a special gates named $g_0, \ldots, g_{a-1}$, with g_i outputs 1 iff $X(x)$ holds, for $0 \leq x < a$. Each gate g_x is the output of a monotone sub-circuit C_x, whose inputs are from $g_0, \ldots, g_{x-1}$. Note that these gates also provide the unary representation of $|X^{<x}|$, i.e., $|X^{<x}| = 1 + max\{z < x : X(z)\}$.

Notice that φ may contain nested occurrences of Z. Any atomic sub-formula of $\varphi(x, Z)$ that contain nested occurrences of Z must be of the form $Z(t|Z|)$, for some $\mathcal{L}_A^2$–term t. This can be replaced by $\exists u \leq a(u = |Z| \wedge Z(t(u))$. Therefore we can assume without loss of generality that Z does not occur nested in $\varphi(x, Z)$.

Also, using the De Morgan's rules we can push the negations in $\varphi(x, Z)$ so that they appear only in front of the atomic sub-formulas. We use *literal* to refer to either an atomic sub-formula or its negation, of $\varphi(x, Z)$.

Now the monotone sub-circuit C_x is as follows. A part of C_x is the monotone circuit C'_x that computes $\varphi(x, Z)$ given the values of the literals (the output gate of C'_x is g_x). This part is constructed in a straightforward way, e.g., the $\vee$ connectives and $\exists z < t$ quantifiers correspond to $\exists$–gates, etc. It remains to evaluate the literals inputs to C'_x, and this is the remaining part of C_x.

Note that the atomic formulas are of the forms

$$s = t \qquad s < t \qquad Z(t)$$

where s, t may contain $|Z|$. Any literal that does not contain $|Z|$ is either TRUE or FALSE. For others, it suffices to evaluate an arbitrary term $t(|Z|)$, where $|Z|$ is given by a unary string, as noted earlier. This can be done in uniform $\mathbf{TC}^0$, e.g., as shown in [1]. $\square$

5 Conclusions

We introduce the two–sorted first–order theories **VL**, **VSL**, **VNL** and **VP** that characterize **L**, **SL**, **NL** and **P**, respectively. Each of these theories is obtained from $\mathbf{V}^0$ by adding an axiom that encodes a polytime algorithm for solving the complete problem of the corresponding class. Here our choices of the complete problems are the *st*–Connectivity Problem and the Circuit Value Problem. Our theories are finitely axiomatizable because $\mathbf{V}^0$ is.

We prove the characterization of the classes by following the method developed in [5]. Here we introduce universal theories that are conservative extensions of the original theories, and show that these universal theories characterize the corresponding classes. (Note that in [5], **VPV** is also a universal theory over the language of polytime functions. The universal theory $\overline{\mathbf{VP}}$ defined in the same way that $\overline{\mathbf{VNL}}$ is defined (Definition 3.8) has a different style.)

Our theories are "minimal" in the sense that they have universal, conservative extensions over the language of the functions in the corresponding classes. Nevertheless, we show that $\mathbf{VL} = \mathbf{\Sigma}_0^B$-**Rec** and $\mathbf{VP} = \mathbf{TV}^0$. And it has been shown recently that $\mathbf{VNL} = \mathbf{V}^1$-**KROM** [12]. Similar arguments would show that $\mathbf{VSL} = \mathbf{V}^1$-**SymKROM**.

An issue that we have not discussed is the connections between our theories and the propositional proof systems, i.e., the propositional translation of the proofs in our theories. This is a subject of further investigation.

6 Acknowledgments

I benefit greatly from discussions with Steve Cook. I would like also to thank him for reading a draft of this paper.

References

[1] Maria Luisa Bonet, Toniann Pitassi, and Ran Raz. On Interpolation and Automatization for Frege Systems. *SIAM Journal on Computing*, 29(6):1939–1967, 2000.

[2] Samuel Buss. *Bounded Arithmetic*. Bibliopolis, Naples, 1986.

[3] Peter Clote and Gaisi Takeuti. Bounded Arithmetic for NC, ALOGTIME, L and NL. *Annals of Pure and Applied Logic*, 56:73–117, 1992.

[4] Stephen Cook. Proof Complexity and Bounded Arithmetic. Course Notes for CSC 2429S. `http://www.cs.toronto.edu/~sacook/`.

[5] Stephen Cook. Theories for Complexity Classes and Their Propositional Translations. manuscript.

[6] Stephen Cook. Feasibly Constructive Proofs and the Propositional Calculus. In *Proceedings of the 7th Annual ACM Symposium on the Theory of Computing*, 1975.

[7] Stephen Cook and Antonina Kolokolova. A Second-order System for Polytime Reasoning Based on Grädel's Theorem. *Annals of Pure and Applied Logic*, pages 193–231, 2003.

[8] Stephen Cook and Antonina Kolokolova. A Second-order Theory for NL. In *Logic in Computer Science (LICS)*, 2004.

[9] Erich Grädel". Capturing complexity classes by fragments of second order logic. *Theorectical Computer Science*, 101:35–57, 1992.

[10] Neil Immerman. Nondeterministic space is closed under complementation. *SIAM J. Comput.*, 17(5):935–938, 1988.

[11] Neil Immerman. *Descriptive Complexity*. Springer, 1999.

[12] Antonina Kolokolova. *Systems of Bounded Arithmetic from Descriptive Complexity*. PhD thesis, University of Toronto, 2004.

[13] R. Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Informatica*, 26:279–284, 1988.

[14] Domenico Zambella. End Extensions of Models of Linearly Bounded Arithmetic. *Annals of Pure and Applied Logic*, 88:263–277, 1997.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'