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

Phuong Nguyen

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 S2 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 $\Sigma^0_2$-Rec, VP is the same as Cook’s $TV_0$, and VNL and VSL are respectively the same as $V^1$-KROM [8] and Kočokolova’s $V^1$-SymKROM [12]. Except for $VL = \Sigma^0_2$-Rec, establishing these equivalences is non-trivial.

1 Introduction

We study the logical characterization of the complexity classes L, SL, NL and P. The first three are the classes of languages computable by respectively: deterministic, symmetric non-deterministic, and non-deterministic 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 T in the sense that the provably total functions in T are precisely the functions in FC. For example there have been a number of theories that characterize P: $S^1_2$ [2], $V^1$-HORN [7], $V^1$, $TV^0$ [5] and VP [6]. PV [5]. The theory $S^1_2$ is the first level in the hierarchy $S^1_2 \subseteq S^2_2 \subseteq \ldots$ of single-sorted first-order theories which characterize the polynomial time hierarchy. The theory $V^1$ is the two-sorted version of $S^1_2$, while $V^1$-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 $TV^0$ (which has been shown equivalent to $V^1$-HORN) is a finitely axiomatizable theory which can be seen as minimal for polynomial-time reasoning. (The two theories $V^1$ and $TV^0$ lie in the hierarchy of two-sorted theories $V^0 \subseteq TV^0 \subseteq V^1 \subseteq TV^1 \subseteq \ldots$ which we will not discuss here.)

In case of NL, the second-order theory $S^{NL\log}_2$ [3] is defined by formulating computation of logspace Turing machines. The theory $V^1$-KROM [8], on the other hand, is similar to $V^1$-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 $V^1$-KROM, the Horn formulas are replaced by the Krom formulas.

Similar to $V^1$-KROM, the theory $V^1$-SymKROM has the same style as $V^1$-HORN. It is defined in the same way as $V^1$-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 $V^1$-HORN, $V^1$-KROM and $V^1$-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 $TV^0$, and thus exhibits a finite axiomatization of $TV^0$. For L, the theory VL for L obtained by our method, not surprisingly, is equivalent to Zambella’s theory $\Sigma^0_2$-Rec [14]. Consequently, it demonstrates the finite axiomatizability of $\Sigma^0_2$-Rec. For NL, our theory VNL seems more natural than $V^1$-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 $V^1$-KROM and VNL are the same. With similar arguments one can see that $V^1$-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 VNL of VNL in detail. From the fact that VNL is a universal theory and a conservative extension of VNL, by applying the Herbrand’s Theorem for VNL we obtain the Witnessing Theorem for VNL: Any \( \Sigma_1 \) 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 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 S^{Log} and 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 S^1_2 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 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 $\text{AC}^0$. Similar arguments show that $\text{USTCONN}$ and $\text{1-STCONN}$ are hard for $\text{SL}$ and $\text{L}$, 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 $t$ if 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 $k = 0 .. (a - 1)$ 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 $V^0$ 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

$$L^2_A = \{0, 1, +, \cdot, |; \vdash, =, <, \leq, \in\}$$

and (ii) the comprehension axiom scheme over $\Sigma^B_0$ formulas, $\Sigma^B_0$-COMP. Here $\Sigma^B_0$ is the set of all formulas whose quantifiers are bounded and are over numbers only. Then $\Sigma^B_0$-COMP is the set of all formulas of the form

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

for $\phi$ a $\Sigma^B_0$ formula, and $X$ does not occur free in $\phi$.

Other important sets of formulas include $\Sigma^B_1$ and $\Pi^B_1$, the sets of formulas of the form $\exists \bar{X} \leq \bar{\nu} \varphi(\bar{X})$ and $\forall \bar{X} \leq \bar{\nu} \varphi(\bar{X})$ respectively, where $\varphi(\bar{X})$ is a $\Sigma^B_0$ formula.

We often use $=_{1}$ 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 $(x, y)$ for the pairing function:

$$(x, y) = (x + y)(x + y + 1) + x$$
We will simply use $Z(x, y)$ for $Z((x, y))$. 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

$$|Z[k]| \leq |Z| \land Z[k](x) \leftrightarrow Z(k, x)$$

Each of our theories is axiomatized by $V^0$ 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)**

$$LC(a, E, Z) \equiv |Z| \leq (a, a) \land Z(0, 0) \land \forall i < a, 0 < i \Downarrow \lnot Z(0, i) \land \forall k, i < a, Z(k + 1, i) \leftrightarrow [Z(k, i) \lor \exists j < a, E(j, i) \land Z(k, j)].$$

(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 $V^0$. Nevertheless, $V^0$ proves that if such $Z$ exists, it is unique.

**Lemma 2.2** $V^0 \vdash (LC(z, E, Z_1) \land LC(z, E, Z_2)) \supset Z_1 = Z_2$

**Proof** The Lemma follows easily from the fact that $V^0$ proves the number induction scheme on $\Sigma^B_0$ formulas, i.e., the set of all formulas of the form

$$\phi(0) \land \forall x < y (\phi(x) \supset \phi(y)) \supset \phi(y)$$

where $\phi$ is a $\Sigma^B_0$ formula. \hfill $\square$

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

**Definition 2.3 (VL, VSL, VNL)**

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

(2)

$$VSL =_{\text{def}} V^0 + [\forall i, j < a E(i, j) \Leftrightarrow E(j, i)] \supset \exists Z \, LC(a, E, Z),$$

(3)

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

(4)

Note that the length of $Z$ is bounded by $(a, a)$. Therefore the theories are bounded. Also, $V^0$ is finitely axiomatizable [7]. Therefore $VL, VSL$ and $VNL$ are all finitely axiomatizable.

**Corollary 2.4** The theories $VL, VSL$ and $VNL$ are finitely axiomatizable.

The theory $\Sigma^B_0$-Rec introduced by Zambella [14] is $V^0$ 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

$$\forall i < a \exists j < a \phi(i, j) \supset \exists Z, fval(0, Z) = 0 \land \forall w < a \phi(fval(w, Z), fval(w + 1, Z)).$$

(5)
where \( \varphi \) is a \( \Sigma^B_0 \) formula not involving \( Z \).

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

\[
\text{val}(x, Z) = z \leftrightarrow Z(x, z) \land \forall y < |Z|, \ Z(x, y) \supset y = z.
\]

It is not surprising that \( \text{VL} \) is the equivalent to \( \Sigma^B_0\text{-Rec} \), and thus \( \text{VL} \) exhibits a finite axiomatization of \( \Sigma^B_0\text{-Rec} \).

**Lemma 2.5** \( \text{VL} = \Sigma^B_0\text{-Rec} \).

**Corollary 2.6** \( \Sigma^B_0\text{-Rec} \) is finitely axiomatizable.

**Proof of Lemma 2.5**

a) \( \text{VL} \subseteq \Sigma^B_0\text{-Rec} \): Suppose that \( E \) is a string such that \( \forall i < a \exists j < a E(i, j) \). We need to prove in \( \Sigma^B_0\text{-Rec} \) that there exists \( Z \) such that \( \text{LC}(a, E, Z) \) 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 \( \text{val}(w, Z') \) 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 \( \text{LC}(a, E, Z) \), 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 \( \Sigma^B_0\text{-COMP} \) as follows.

\[
Z(i, j) \leftrightarrow \exists w \leq i, j = \text{val}(w, Z').
\]

b) \( \Sigma^B_0\text{-Rec} \subseteq \text{VL} \): Let \( \varphi(i, j) \) be a \( \Sigma^B_0 \) formula such that \( \forall i < a \exists j < a \varphi(i, j) \). We need to prove in \( \text{VL} \) 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 \( \Sigma^B_0\text{-COMP}, \text{VL}^0 \) proves that

\[
\exists E < \langle a, a \rangle, \ E(i, j) \leftrightarrow i < a \land j < a \land \varphi(i, j) \land \forall \ell < j \neg \varphi(i, \ell).
\]

Let \( Z' \) be the string that satisfies \( \text{LC}(a, E, Z') \). 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:

\[
\text{val}(w, Z) = z \leftrightarrow Z'(w, z) \land \forall w' < w \neg Z(w', z).
\]

Hence in \( \text{VL}^0 \), \( Z \) can be defined using \( \Sigma^B_0\text{-COMP} \) by:

\[
|Z| \leq \langle a, a \rangle \land [\forall w, z < a, Z(w, z) \leftrightarrow Z'(w, z) \land \forall w' < w \neg Z(w', z)]
\]

2.3 \( \text{VP} \)

Now we will define the theory \( \text{VP} \) for \( \text{P} \) using \( \text{CVP} \), 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

- \( \lor \)-gates that have at least one input from the gates we have in loop \( k \);
- \( \land \)-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 \( k = 0 \ldots (a - 1) \) do
   - \( Z^{[k + 1]} := Z^{[k]} \)
   - For \( 0 \leq u < a \)
     - if \( u \) is an \( \land \)-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 \( \lor \)-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 \( \text{CVP} \)

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 \( \land \)-gates are labeled by the even numbers, and the \( \lor \)-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)**

\[
ALC(a, E, Z) \equiv |Z| \leq \langle a, a \rangle \land Z(0, 1) \land \forall i < a \ i \neq 1 \lor \lnot Z(0, i) \land \\
\forall i, k < a, \ Z(k + 1, 2i) \leftrightarrow [Z(k, 2i) \lor \forall j < a \ E(2i, j) \lor Z(k, j)] \land \\
Z(k + 1, 2i + 1) \leftrightarrow [Z(k, 2i + 1) \lor \exists j < a \ E(2i + 1, j) \land Z(k, j)]
\]

(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) \( \text{VP} = \text{def} \ V^0 + \exists Z ALC(a, E, Z) \).**

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

Note that \( \text{P} \) 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 $\Sigma^B_1$ and $\Pi^B_1$ formulas are of the form $\exists \vec{X} \leq \vec{\varphi}(\vec{X})$ and $\forall \vec{X} \leq \vec{\varphi}(\vec{X})$, respectively, where $\varphi$ is a $\Sigma^B_1$ formula. We say that a relation $R(\vec{x}, \vec{X})$ is $\Delta^B_1$-definable in a theory $T$ if there are a $\Sigma^B_1$ formula $\varphi(\vec{x}, \vec{X})$ and a $\Pi^B_1$ formula $\psi(\vec{x}, \vec{X})$ such that

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

Also note that by Parikh’s Theorem, for each of our theories the class of $\Sigma_1$-definable functions is the same as the class of $\Sigma^B_1$-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 $\Sigma_1$-definable functions of VC, and the relations in C are precisely the $\Delta^B_1$-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 $\Delta^B_1$ 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 $AC^0$ relation (or equivalently, a $\Sigma^B_1$ formula) $\text{STEP}_M(\vec{x}, \vec{X}, i, j)$ such that $\text{STEP}_M(\vec{x}, \vec{X}, i, j)$ 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 $\text{STEP}_M$ 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 $L_{\Sigma_1}$ number term $\text{nconf}_{M}(\vec{x}, \vec{X})$ that bounds the number of different configurations of $M$ on input $(\vec{x}, \vec{X})$ where $|\vec{X}| = n$ (we write $t_M$ for $\text{nconf}_{M}(\vec{x}, |\vec{X}|)$).

**Definition 3.2** For each non-deterministic logspace Turing machine $M$, $\text{GRAPH}_M(\vec{x}, \vec{X}, E)$ is the following $\Sigma^B_1$ formula:

$$\text{GRAPH}_M(\vec{x}, \vec{X}, E) \equiv |E| \leq (t_M, t_M) \land \forall i, j < t_M \ E(i, j) \iff \text{STEP}_M(\vec{x}, \vec{X}, i, j).$$

(7)

And $\text{Graph}_M(\vec{x}, \vec{X})$ is the $AC^0$ function whose graph is $\text{GRAPH}_M$:

$$\text{Graph}_M(\vec{x}, \vec{X})(i, j) \iff i < t_M \land j < t_M \land \text{STEP}_M(\vec{x}, \vec{X}, i, j)$$

(8)
Then, \( R(\bar{x}, |\bar{x}|) \) is represented by the following \( \Sigma^B \) and \( \Pi^B \) formulas

\[
R(\bar{x}, |\bar{x}|) \iff \exists E, Z, \text{ GRAPH}_M(\bar{x}, \bar{x}, E) \land \text{LC}(t, E, Z) \land Z(t, 1) \tag{9}
\]
\[
\forall E, Z, (\text{ GRAPH}_M(\bar{x}, \bar{x}, E) \land \text{LC}(t, E, Z)) \implies Z(t, 1) \tag{10}
\]

(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 \text{VNL}.

(9) \implies (10) \( E \) is unique because \( E = \text{Graph}_M(\bar{x}, \bar{x}) \). The uniqueness of \( Z \) follows from the uniqueness of \( E \) and Lemma 2.2.

(10) \implies (9) \( E \) exists because \( E = \text{Graph}_M(\bar{x}, \bar{x}) \). \( Z \) exists using \text{LC} (1).

\[\square\]

**Theorem 3.3** The NL functions are \( \Sigma^B \) definable in \text{VNL}.

Before proving this theorem, we prove the following lemma, which states that \text{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 \((z, \bar{x}, \bar{x})\),

\[
\text{VNL} \vdash \exists ! E_1, Z_1 \forall z < a, [\text{GRAPH}_M(z, \bar{x}, \bar{x}, E^{[1]}_1) \land \text{LC}(a, E^{[1]}_1, Z^{[1]}_1)] \tag{11}
\]

In other words, for each value of \( z \), \( E^{[1]}_1 \) is the graph of computations of \( M \) on input \((z, \bar{x}, \bar{x})\), and \( Z^{[1]}_1 \) is the reachability relation of \( E^{[1]}_1 \). \( Z^{[1]}_1(k, i) \) holds if and only if there is a path of length \( \leq k \) from 0 to \( i \) in the graph \( E^{[1]}_1 \).

**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, \( E^{[1]}_1 \) is just \( \text{Graph}_M(z, \bar{x}, \bar{x}) \), therefore \( E_1 \) can be defined in \text{V}^0 \ using \( \Sigma^B \text{-COMP} \), i.e.,

\[
E_1(z, i, j) \iff z < a \land i < a \land j < a \land \text{Graph}(z, \bar{x}, \bar{x})(\langle i, j \rangle)
\]

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 \( E^{[1]}_1 \). 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” \( E^{[1]}_1 \), 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 \( E^{[1]}_1 \) 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 \( E^{[0]}_1 \): there is a path of length \( k \) in \( E^{[0]}_1 \) 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 \( E^{[1]}_1 \) 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

\[
[\forall z, i, j < a, E'(az + i, az + j) \leftrightarrow E^{[1]}_1(i, j)] \land [\forall z < a, E'(0, az)].
\]

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

\[
Z^{[0]}_1(k, i) \leftrightarrow Z'(k, i) \quad \text{and} \quad Z^{[1]}_1(k, i) \leftrightarrow Z'(k + 1, az + i), \text{ for } 1 \leq z < a.
\]

Since \text{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 $F(\vec{z}, \vec{x})$ 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 $a = \max\{s, nconf_M(\vec{z}, \vec{x})\}$. Then for each “bit $z$”, $z \in F(\vec{z}, \vec{x})$ iff $M$ accepts the input $(\vec{z}, \vec{x})$, i.e., in the “small graph” $E_i^3$ there is a path from the source 0 to the target 1. In other words, $F$ can be defined as follows

$$F(\vec{z}, \vec{x}) = Y \iff \forall z \leq s \ [Y(z) \iff Z_i^3(a, 1)].$$

It follows that VNL can $\Sigma^b_1$ define $F$.

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

$$y = f(\vec{z}, \vec{x}) \iff Z_i^1(a, 1)$$

Thus $f$ is $\Sigma_1$-definable in VNL. 

\[\square\]

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

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

$$\text{STCONN}(x, E) = Z \iff LC(x, E, Z).$$

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

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

$$F_{a,t}(\vec{z}, \vec{x})(z) \iff z < t \land \alpha(z, \vec{z}, \vec{x}) \quad (13)$$

$$f_{a,t}(\vec{z}, \vec{x}) \in t(\vec{x}, \vec{x}) \quad (14)$$

$$f_{a,t}(\vec{z}, \vec{x}) < t \lor \alpha(f_{a,t}(\vec{z}, \vec{x}), \vec{x}, \vec{x}) \quad (15)$$

$$z < f_{a,t}(\vec{z}, \vec{x}) \lor \neg \alpha(z, \vec{z}, \vec{x}) \quad (16)$$

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

a) it is represented by an open $\mathcal{L}_{\text{NL}}$ formula;

b) it is represented by a $\Sigma^b_0(\mathcal{L}_{\text{NL}})$ formula.

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

$$R(\vec{z}, \vec{x}) \iff \text{STCONN}(t, \text{Graph}_M(\vec{z}, \vec{x}))(t, 1),$$

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

9
It remains to show that any $\Sigma_0^b(\mathcal{L}_{NL})$ formula $\varphi(\vec{x}, \vec{X})$ can be evaluated by $\mathsf{NL}$ machines. This is proved by induction on the structure of $\varphi$. The base case is straightforward. For the induction step, we need to use the fact that $\mathsf{NL}$ is closed under (i) negation, (ii) Boolean operations, and (iii) bounded number quantifications. Note that (i) follows from the Immermann-Szepesvári result \cite{10, 13}, and (ii) and (iii) are straightforward. \hfill \Box

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

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

$$F(\vec{x}, \vec{X})(i) \iff i < p \land R(i, \vec{x}, \vec{X})$$

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

The case of $\mathsf{NL}$ number function is similar. \hfill \Box

### 3.3 The Open Theory $\mathcal{VNL}$ and The Witnessing Theorem for $\mathcal{VNL}$

Note that $\mathcal{L}C$ (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 $\mathcal{L}C$ can be eliminated using the following technique from \cite{5}. Let $f_{LC}$ be the $\mathsf{AC}^0$ function defined by

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

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

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

Therefore we obtain the following open formula (in $\mathcal{L}C_3 \cup \{f_{LC}\}$) which is equivalent to $\mathcal{L}C$:

$$\mathcal{L}C'(a, E, Z) \equiv Z(0, 0) \land [i < a \supset Z(0, i)] \land [k < a \land i < a \supset [Z(k, i) \leftrightarrow (Z(k, i) \lor f_{LC}(k, i, a, E, Z) < a)]. \quad (17)$$

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

**Lemma 3.9** 1. For each $\Sigma_0^b(\mathcal{L}_{NL})$ formula $\varphi$, there is an open $\mathcal{L}_{NL}$-formula $\varphi'$ such that

$$\mathcal{VNL} \vdash \varphi' \iff \varphi$$

2. $\mathcal{VNL} \vdash \Sigma_0^b(\mathcal{L}_{NL})$-COMP.

**Proof** 1) is proved by structural induction on $\varphi$. The case of quantifier in the induction step is proved by the same method that we have used to eliminate the quantifier in $\mathcal{L}C$.

2) follows from 1) and the fact that

$$Z = F_{a, a} \iff |Z| \leq t \land \forall z < tZ(z) \iff a(z)$$

(See (13).) \hfill \Box
Lemma 3.10 \( \text{VNL} \) is a conservative extension of \( \text{VNL} \).

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

It remains to show that \( \text{VNL} \) is conservative over \( \text{VNL} \). It suffices to show that \( \text{VNL} \vdash \Sigma^B_0(\mathcal{L}_{\text{NL}})\)-COMP. Note that the functions of \( \mathcal{L}_{\text{NL}} \) are \( \Sigma_1 \)-definable in \( \text{VNL} \) (Theorem 3.3). Consider a \( \Sigma^B_0(\mathcal{L}_{\text{NL}}) \) formula \( \varphi(z) \). By Lemma 3.6 it represents an NL relation, i.e., the bit graph of an FNL function \( F \). By Theorem 3.3, \( F \) is \( \Sigma^B_0 \)-definable in \( \text{VNL} \), thus \( \text{VNL} \) proves the comprehension axiom for \( \varphi(z) \). \( \square \)

Theorem 3.11 (Witnessing Theorem for \( \text{VNL} \)) \( \Sigma^B_0 \) theorems of \( \text{VNL} \) are witnessed by NL functions.

Proof Suppose that \( \exists Y \varphi(x, X) \) is a \( \Sigma^B_0 \) theorem of \( \text{VNL} \). Then by Lemma 3.10, it is also a theorem of \( \text{VNL} \). Since \( \text{VNL} \) is a universal theory, by the Herbrand Theorem, there is a function of \( \mathcal{L}_{\text{NL}} \) such that \( \text{VNL} \vdash \varphi(x, X) \). \( \square \)

Corollary 3.12 A function is in FNL if and only if it is \( \Sigma^B_0 \)-definable in \( \text{VNL} \). A relation is in NL if and only if it is \( \Delta^B_0 \)-definable in \( \text{VNL} \).

4 \( \text{VP} = \text{TV}^0 \)

In [5] Cook introduces the theory \( \text{TV}^0 \), and shows that it characterizes \( \text{P} \). He also shows that \( \text{TV}^0 \) is equivalent to \( \text{V}^0 + \Sigma^B_0 \text{-BIT-REC} \), where \( \Sigma^B_0 \text{-BIT-REC} \), the bit-recursion scheme for \( \Sigma^B_0 \) formulas, is the following scheme:

\[
\exists X \leq a \forall x < a, \; X(x) \leftrightarrow \varphi(x, X^{<x}),
\]

where \( \varphi \) is \( \Sigma^B_0 \), and \( X^{<x} \) is the \( \text{AC}^0 \) function defined by

\[
X^{<x}(z) \leftrightarrow z < x \land X(z).
\]

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

Theorem 4.1 \( \text{VP} = \text{TV}^0 \).

Corollary 4.2 ([5]) \( \text{TV}^0 \) 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 \( \Sigma^B_0 \text{-BIT-REC} \) axioms. Hence \( \text{VP} \) is a sub-theory of \( \text{TV}^0 \). It remains to show that the \( \Sigma^B_0 \text{-BIT-REC} \) axioms hold in \( \text{VP} \).

Let \( \varphi(x, Z) \) be a \( \Sigma^B_0 \) formula. We need to prove in \( \text{VP} \) 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 \( \varphi \) 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 \( \mathcal{L}_A \)-term \( t \). This can be replaced by \( \exists u \leq a(u = [Z] \land Z(t(u))) \). 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 \( C_z \) is as follows. A part of \( C_z \) is the monotone circuit \( C^*_z \) that computes \( \varphi(x, Z) \) given the values of the literals (the output gate of \( C^*_z \) is \( g_z \)). This part is constructed in a straightforward way, e.g., the \( \lor \) connectives and \( \exists < t \) quantifiers correspond to \( \exists \)-gates, etc. It remains to evaluate the literals inputs to \( C^*_z \), and this is the remaining part of \( C_z \).

Note that the atomic formulas are of the forms

\[
s = t \quad s < t \quad Z(t)
\]

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 \( TC^0 \), e.g., as shown in [1].

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 \( V^0 \) 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 \( V^0 \) 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 \( VF \) defined in the same way that \( VNL \) 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 \( VL = \Sigma^P_0 \)-Rec and \( VP = TV^0 \). And it has been shown recently that \( VNL = V^{1-}\text{KROM} \) [12]. Similar arguments would show that \( VSL = V^{1-}\text{SymKROM} \).

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


