Relativizing Small Complexity Classes and their Theories

Klaus Aehlig* Stephen Cook Phuong Nguyen

May 30, 2007

Abstract

Existing definitions of the relativizations of $\mathbf{NC}^1$, $\mathbf{L}$ and $\mathbf{NL}$ do not preserve the inclusions $\mathbf{NC}^1 \subseteq \mathbf{L}$, $\mathbf{NL} \subseteq \mathbf{AC}^1$. We start by giving the first definitions that preserve them. Here for $\mathbf{L}$ and $\mathbf{NL}$ we define their relativizations using Wilson's stack oracle model, but limit the height of the stack to a constant (instead of $\log(n)$). We show that the collapse of any two classes in $\{\mathbf{AC}^0(m), \mathbf{TC}^0, \mathbf{NC}^1, \mathbf{L}, \mathbf{NL}\}$ implies the collapse of their relativizations. Next we develop theories that characterize the relativizations of subclasses of $\mathbf{P}$ by modifying theories previously defined by the second two authors. A function is provably total in a theory iff it is in the corresponding relativized class. Finally we exhibit an oracle α that makes $\mathbf{AC}^k(\alpha)$ a proper hierarchy. This strengthens and clarifies the separations of the relativized theories in [Takeuti, 1995]. The idea is that a circuit whose nested depth of oracle gates is bounded by k cannot compute correctly the $(k+1)$ compositions of every oracle function.

1 Introduction

Oracles that separate $\mathbf{P}$ from $\mathbf{NP}$ and oracles that collapse $\mathbf{NP}$ to $\mathbf{P}$ have both been constructed. This rules out the possibility of proofs of the separation or collapse of $\mathbf{P}$ and $\mathbf{NP}$ by methods that relativize. However, similar results have not been established for subclasses of $\mathbf{P}$ such as $\mathbf{L}$ and $\mathbf{NL}$. Indeed, prior to this work there has not been a satisfying definition of the relativized version of $\mathbf{NL}$ that preserves simultaneously the inclusions

$$\mathbf{NC}^1 \subseteq \mathbf{L} \subseteq \mathbf{NL} \subseteq \mathbf{AC}^1 \tag{1}$$

For example [LL76], if the Turing machines are allowed to be nondeterministic when writing oracle queries, then there is an oracle α so that $\mathbf{NL}(\alpha) \not\subseteq \mathbf{P}(\alpha)$. Later definitions of $\mathbf{NL}(\alpha)$ adopt the requirement specified in [RST84] that the nondeterministic oracle machines be deterministic whenever the oracle tape (or

*supported by DFG grant Ae 102/1-1

oracle stack) is nonempty. Then the inclusion $\mathbf{NL}(\alpha) \subseteq \mathbf{P}(\alpha)$ relativizes, but not all inclusions in (1).

Because the nesting depth of oracle gates in an oracle $\mathbf{NC}^1$ circuit can be bigger than one, the model of relativization that preserves the inclusion $\mathbf{NC}^1 \subseteq \mathbf{L}$ must allow an oracle logspace Turing machine to have access to more than one oracle query tape [Orp83, Bus86, Wil88]. For the model defined by Wilson [Wil88], the partially constructed oracle queries are stored in a stack. The machine can write queries only on the oracle tape at the top of the stack. It can start a new query on an empty oracle tape (thus *pushing* down the current oracle tape, if there is any), or query the content of the top tape which then becomes empty and the stack is *popped*.

Following Cook [Coo85], the circuits accepting languages in relativized $\mathbf{NC}^1$ are those with logarithmic depth where the Boolean gates have bounded fanin and an oracle gate of m inputs contributes $\log(m)$ to the depths of its parents. Then in order to relativize the inclusion $\mathbf{NC}^1 \subseteq \mathbf{L}$, the oracle logspace machines defined by Wilson [Wil88] are required to satisfy the condition that at any time,

$$\sum_{i=1}^{k} max\{\log(|q_i|), 1\} = \mathcal{O}(\log(n))$$

where $q_1, q_2, \ldots, q_k$ are the contents of the stack and $|q_i|$ are their lengths. For the simulation of an oracle $\mathbf{NC}^1$ circuit by such an oracle logspace machine the upper bound $\mathcal{O}(\log(n))$ cannot be improved.

Although the above definition of $\mathbf{L}(\alpha)$ (and $\mathbf{NL}(\alpha)$) ensures that $\mathbf{NC}^1(\alpha) \subseteq \mathbf{L}(\alpha)$, unfortunately we know only that $\mathbf{NL}(\alpha) \subseteq \mathbf{AC}^2(\alpha)$ [Wil88]; the inclusion $\mathbf{NL}(\alpha) \subseteq \mathbf{AC}^1(\alpha)$ is left open.

We observe that if the height of the oracle stack is bounded by a constant (while the lengths of the queries are still bounded by a polynomial in the length of the inputs), then an oracle $\mathbf{NL}$ machine can be simulated by an oracle $\mathbf{AC}^1$ circuit, i.e., $\mathbf{NL}(\alpha) \subseteq \mathbf{AC}^1(\alpha)$. In fact, $\mathbf{NL}(\alpha)$ can then be shown to be the $\mathbf{AC}^0(\alpha)$ closure of the Reachability problem for directed graphs. Similarly, $\mathbf{L}(\alpha)$ is the $\mathbf{AC}^0(\alpha)$ closure of the Reachability problem for directed graphs whose outdegree is at most one.

The $\mathbf{AC}^0(\alpha)$ closure of the Boolean Sentence Value problem (which is $\mathbf{AC}^0$ complete for $\mathbf{NC}^1$) turns out to be the languages computable by uniform oracle $\mathbf{NC}^1$ circuits (defined as before) where the nesting depth of oracle gates is now bounded by a constant. We redefine $\mathbf{NC}^1(\alpha)$ using this new restriction on the oracle gates; the new definition is more suitable in the context of $\mathbf{AC}^0(\alpha)$ reducibility (the previous definition of $\mathbf{NC}^1(\alpha)$ seems suitable when one considers $\mathbf{NC}^1(\alpha)$ reducibility). Consequently, we obtain the first definition of $\mathbf{NC}^1(\alpha)$, $\mathbf{L}(\alpha)$ and $\mathbf{NL}(\alpha)$ that preserves the inclusions in (1).

Furthermore, the $\mathbf{AC}^0$-complete problems for $\mathbf{NC}^1$, $\mathbf{L}$, and $\mathbf{NL}$ (as well as $\mathbf{AC}^0(m)$, $\mathbf{TC}^0$) become $\mathbf{AC}^0(\alpha)$-complete for the corresponding relativized classes. Therefore the existence of any oracle that separates two of the mentioned classes implies the separation of the respective nonrelativized classes. (If the non-relativised classes would be equal, their complete problems would be equivalent under $\mathbf{AC}^0$-reductions, hence even more under $\mathbf{AC}^0(\alpha)$-reductions and therefore the relativised classes would coincide as well.) Separating the relativized classes is as hard as separating their nonrelativized counterparts. This nicely generalizes known results [Wil88, Sim77, Wil89].

On the other hand, oracles that separate the classes $\mathbf{AC}^k$ (for $k = 0, 1, 2, \ldots$) and $\mathbf{P}$ have been constructed [Wil89]. Here we prove a sharp separation between relativized circuit classes whose nesting depths of oracle gates differ by one. More precisely, we show that a family of uniform circuits with nesting depth of oracle gates bounded by k cannot compute correctly the $(k+1)$ iterated compositions

$$f(f(\ldots f(0)\ldots)) \tag{2}$$

for all oracle function f. (Clearly (2) can be computed correctly by a circuit with oracle gates having nesting depth $(k+1)$.) As a result, there is an oracle α such that

$$\mathbf{NL}(\alpha) \subsetneq \mathbf{AC}^1(\alpha) \subsetneq \mathbf{AC}^2(\alpha) \subsetneq \ldots \subsetneq \mathbf{P}(\alpha) \tag{3}$$

The idea of using (2) to separate relativized circuit classes is already present in the work of Takeuti [Tak95] where it is used to separate the relativized versions of first-order theories $TLS(\alpha)$ and $TAC^1(\alpha)$. Here TLS and TAC are (single sorted) theories associated with $\mathbf{L}$ and $\mathbf{AC}^1$, respectively. Thus with simplified arguments we strengthen his results.

Finally, building up from the work of the second two authors [CN06, NC05] we develop relativized two-sorted theories that are associated with the newly defined classes $\mathbf{NC}^1(\alpha), \mathbf{L}(\alpha), \mathbf{NL}(\alpha)$ as well as other relativized circuit classes.

The paper is organized as follows. In Section 2 we define the relativized classes and prove the inclusions mentioned above. In Section 3 we define the associated theories. An oracle that separates classes in (3) is shown in Section 4.

2 Definitions of Small Relativized Classes

2.1 Relativized Circuit Classes

Throughout this paper, α denotes a unary relation on binary strings.

A problem is in $\mathbf{AC}^k$ if it can be solved by a polynomial size family of Boolean circuits whose depth is bounded by $\mathcal{O}((\log n)^k)$ (n is the number of the inputs), where $\wedge$ and $\vee$ gates are allowed unbounded fanin. The relativized class $\mathbf{AC}^k(\alpha)$ generalizes this by allowing, in addition to (unbounded fanin) Boolean gates $(\neg, \wedge, \vee)$, oracle gates that output 1 if and only if the inputs to the gates (viewed as binary strings) belong to α (these gates are also called α gates).

In this paper we always require circuit families to be *uniform*. Our default definition of uniform is DLOGTIME, a robust notion of uniformity that has a number of equivalent definitions [BIS90, Imm99]. In particular, a language $L \subseteq \{0,1\}^*$ is in (uniform) $\mathbf{AC}^0$ iff it represents the set of finite models $\{1, \ldots, n\}$ of some fixed first-order formula with an uninterpreted unary predicate symbol and ternary predicates which are interpreted as addition and multiplication.

Recall that $\mathbf{TC}^0$ (resp. $\mathbf{AC}^0(m)$) is defined in the same way as $\mathbf{AC}^0$, except the circuits allow unbounded fanin threshold (resp. mod$m$) gates.

Definition 1 ($\mathbf{AC}^k(\alpha)$, $\mathbf{AC}^0(m,\alpha)$, $\mathbf{TC}^0(\alpha)$)**.** *For $k \geq 0$, the class $\mathbf{AC}^k(\alpha)$ (resp. $\mathbf{AC}^0(m,\alpha)$, $\mathbf{TC}^0(\alpha)$) is defined as uniform $\mathbf{AC}^k$ (resp. $\mathbf{AC}^0(m)$, $\mathbf{TC}^0$) except that unbounded fan-in α gates are allowed.*

The class $\mathbf{NC}^k$ is the subclass of $\mathbf{AC}^k$ defined by restricting the $\wedge$ and $\vee$ gates to have fanin 2. Defining $\mathbf{NC}^k(\alpha)$ is more complicated. In [Coo85] the depth of an oracle gate with m inputs is defined to be $\log(m)$. A circuit is an $\mathbf{NC}^k(\alpha)$-circuit provided that it has polynomial size and the total depth of all gates along any path from the output gate to an input gate is $\mathcal{O}((\log n)^k)$. Note that if there is a mix of large and small oracle gates, the number of oracle gates may not be $\mathcal{O}((\log n)^{k-1})$.

Here we restrict the definition further, requiring that the nested depth of oracle gates is $\mathcal{O}((\log n)^{k-1})$.

Definition 2 ($\mathbf{NC}^k(\alpha)$)**.** *For $k \geq 1$, a language is in $\mathbf{NC}^k(\alpha)$ if it is computable by a uniform family of $\mathbf{NC}^k(\alpha)$ circuits, i.e., $\mathbf{AC}^k(\alpha)$ circuits where the $\wedge$ and $\vee$ gates have fanin 2, and the nested depth of α gates is $\mathcal{O}((\log n)^{k-1})$.*

The following inclusions extend the inclusions of the nonrelativized classes:

$$\mathbf{AC}^0(\alpha) \subsetneq \mathbf{AC}^0(2,\alpha) \subsetneq \mathbf{AC}^0(6,\alpha) \subseteq \mathbf{TC}^0(\alpha) \subseteq \mathbf{NC}^1(\alpha) \subseteq \mathbf{AC}^1(\alpha) \subseteq \dots$$

2.2 Relativized Logspace Classes

To define oracle logspace classes, we use a modification of Wilson's stack model [Wil88]. An advantage is that the relativized classes defined here are closed under $\mathbf{AC}^0$-reductions. This is not true for the non-stack model.

A Turing machine M with a stack of oracle tapes can write 0 or 1 onto the top oracle tape if it already contains some symbols, or it can start writing on an empty oracle tape. In the latter case, the new oracle tape will be at the top of the stack, and we say that M performs a *push* operation. The machine can also *pop* the stack, and its next action and state depend on $\alpha(Q)$, where Q is the content of the top oracle tape. Note that here the oracle tapes are write-only.

Instead of allowing an arbitrary number of oracle tapes, we modify Wilson's model by allowing only a stack of constant height (hence the prefix "cs" in cs$\mathbf{L}(\alpha)$ and cs$\mathbf{NL}(\alpha)$). This places the relativized classes in the same order as the order of their unrelativized counterparts.

In the definition of cs$\mathbf{NL}(\alpha)$, we also use the restriction [RST84] that the machine is deterministic when the oracle stack is non empty.

Definition 3 (cs$\mathbf{L}(\alpha)$, cs$\mathbf{NL}(\alpha)$)**.** *For a unary relation α on strings, cs$\mathbf{L}(\alpha)$ is the class of languages computable by logspace, polytime Turing machines using an α-oracle stack whose height is bounded by a constant. cs$\mathbf{NL}(\alpha)$ is defined as cs$\mathbf{L}(\alpha)$ but the Turing machines are allowed to be nondeterministic when the oracle stack is empty.*

Theorem 4. $\mathbf{NC}^1(\alpha) \subseteq cs\mathbf{L}(\alpha) \subseteq cs\mathbf{NL}(\alpha) \subseteq \mathbf{AC}^1(\alpha)$.

Proof. The second inclusion is immediate from the definitions, the first can be proved as in the standard proof of the fact that $\mathbf{NC}^1 \subseteq \mathbf{L}$ (see also [Wil88]). The last inclusion can actually be strengthened, as shown in the next theorem. □

Theorem 5. *Each language in $cs\mathbf{NL}(\alpha)$ can be computed by a uniform family of $\mathbf{AC}^1(\alpha)$ circuits whose nested depth of oracle gates is bounded by a constant.*

Proof. We proceed as in the proof of the fact that $\mathbf{NL} \subseteq \mathbf{AC}^1$. Let M be a nondeterministic logspace Turing machine with a constant-height stack of oracle tapes. Let h be the bound on the height of the oracle stack. There is a polynomial $p(n)$ so that for each input length n and oracle α, M has at most $p(n)$ possible configurations:

$$u_0 = START,\ u_1 = ACCEPT,\ u_2, \ldots, u_{p(n)-1} \tag{4}$$

(Here a configuration u_i encodes information about the state, the content of the work tape, the position of the input tape head and the input symbol being read, but no information about the oracle stack.)

Given an input of length n, consider the directed graph G with vertices (k, u_i) for $0 \le k \le h$, $0 \le i < p(n)$, where the edge relation E is as follows: For u_j a next configuration of u_i,

(i) if M does not *push* or *pop* after u_i, then $((0, u_i), (0, u_j)) \in E$; if furthermore u_i codes a deterministic state, then $((k, u_i), (k, u_j)) \in E$, for $1 \le k \le h$;

(ii) if the next move of M after u_i is *push*, then $((k, u_i), (k+1, u_j)) \in E$ for $0 \le k < h$;

(iii) otherwise, if the move of M after u_i is *pop*, then $((k, u_i), (k-1, u_j)) \in E$ for $1 \le k \le h$.

(Here k is a possible height of the stack when M has configuration u_i.)

Suppose that edge relation E has been computed, then the Reachability relation in G can be computed by an $\mathbf{AC}^1$ circuit. M accepts if and only if $(0, ACCEPT)$ is reachable from $(0, START)$. It remains to show that E can be computed by an $\mathbf{AC}^1(\alpha)$ circuit.

Let E_k denote the subgraph of E that contains the edges in (i,ii), and the edges $((\ell, u_i), (\ell - 1, u_j))$ as in (iii) where $k \le \ell \le h$. (Thus $E_1 = E$.) Also, let E_{h+1} denote the subgraph of E that contains only the edges as in (i,ii).

Note that E_{h+1} can be computed by an $\mathbf{AC}^0$ circuit. We show that E_k can be computed from E_{k+1} by an $\mathbf{AC}^1(\alpha)$ circuit whose oracle depth is one (for $1 \le k \le h$). This will complete our proof of the theorem.

The new edges in E_k are of the form $((k, u_i), (k-1, u_j))$ where u_j is resulted from u_i by a *pop* operation. To check whether u_i, u_j satisfy this condition, we need to compute the oracle query on the current oracle tape that is asked when M moves from u_i to u_j. This query is computed by tracing back the computation of M, starting at u_i, until we hit the first configuration v where the oracle stack height is $k-1$. More precisely, we compute the path in E_{k+1} of the form

$$(k-1, v), (k, v_0), (k_1, v_1), \ldots, (k_t, v_t), (k, u_i)$$

where $k \leq k_1, \ldots, k_t \leq h$. This path can be computed by a deterministic logspace function, and hence an $\mathbf{AC}^1$ circuit.

Now, the oracle query Q asked at u_i can be extracted from the sequence

$$(v, v_0, v_1, \ldots, v_t)$$

by an $\mathbf{AC}^0$ circuit. Then, $((k, u_i), (k-1, u_j)) \in E_k$ if and only if $\alpha(Q)$. □

2.3 cs$\mathbf{L}(\alpha)$ Reducibility

A cs$\mathbf{L}(\alpha)$ function is defined by allowing the cs$\mathbf{L}(\alpha)$ machine to write on a write-only output tape. Then the notion of many-one cs$\mathbf{L}(\alpha)$ reducibility is defined as usual. Using this notion, the next lemma can be used to show that the Immerman-Szelepcsényi Theorem and Savitch's Theorem relativize. Recall that **STCONN** is the problem: given (G, s, t), where s, t are two designated vertices of a directed graph G, decide whether there is a path from s to t.

Lemma 6. *A language is in cs*$\mathbf{NL}(\alpha)$ *iff it is many-one cs*$\mathbf{L}(\alpha)$ *reducible to* **STCONN**.

Proof. The "if" direction is easy, so we prove the "only if" direction. Let $\mathcal{L}$ be a language in cs$\mathbf{NL}(\alpha)$ which is computed by M, an $\mathbf{NL}$ machine with a constant height oracle stack. The cs$\mathbf{L}(\alpha)$ transformation is as follows. Given an input string X to M, the graph G has polynomially many vertices in (4), which are the configurations of M on input X. The edges of G are

(i) (u_i, u_j) where u_j is a next configuration of u_i, and u_i does not write on an empty stack.

(ii) (u_i, u_j) where u_i writes on an empty stack, and u_j is the next time the stack is empty.

The edges in (i) can be computed in $\mathbf{AC}^0$, while the edges in (ii) can be computed in cs$\mathbf{L}(\alpha)$ (because M is deterministic when the oracle stack is non-empty).
□

Corollary 7 (Relativized Immerman-Szelepcsényi Theorem)**.** *cs*$\mathbf{NL}(\alpha)$ *is closed under complementation.*

Proof. Any language in *co*-cs$\mathbf{NL}(\alpha)$ is cs$\mathbf{L}(\alpha)$ reducible to $\overline{\mathbf{STCONN}}$, which is $\mathbf{AC}^0$ reducible to **STCONN**. So *co*-cs$\mathbf{NL}(\alpha) \subseteq$ cs$\mathbf{NL}(\alpha)$. □

Let cs$\mathbf{L}^2(\alpha)$ denote the class of languages computable by a deterministic oracle Turing machine in $\mathcal{O}(\log^2)$ space and constant-height oracle stack.

Corollary 8 (Relativized Savitch's Theorem)**.** *cs*$\mathbf{NL}(\alpha) \subseteq$ *cs*$\mathbf{L}^2(\alpha)$.

Proof. The corollary follows from Lemma 6 and the fact that the composition of a cs$\mathbf{L}(\alpha)$ function and a ($\log^2$) space function (for **STCONN**) is a cs$\mathbf{L}^2(\alpha)$ function. $\square$

3 Relativized Theories

3.1 Two-Sorted Languages and Complexity Classes

Our theories are based on a two-sorted vocabulary, and it is convenient to reinterpret the complexity classes using this vocabulary [CN06, NC05]. Our two-sorted language has variables $x, y, z, ...$ ranging over $\mathbb{N}$ and variables $X, Y, Z, ...$ ranging over finite subsets of $\mathbb{N}$ (interpreted as bit strings). Our basic two-sorted vocabulary $\mathcal{L}_A^2$ includes the usual symbols $0, 1, +, \cdot, =, \leq$ for arithmetic over $\mathbb{N}$, the length function $|X|$ on strings, the set membership relation $\in$, and string equality $=_2$ (where we usually drop mention of the subscript 2). The function $|X|$ denotes 1 plus the largest element in the set X, or 0 if X is empty (roughly the length of the corresponding string). We will use the notation $X(t)$ for $t \in X$, and we will think of $X(t)$ as the t-th bit in the string X.

Number terms of $\mathcal{L}_A^2$ are built from the constants 0,1, variables $x, y, z, ...$, and length terms $|X|$ using $+$ and $\cdot$. The only *string terms* are string variables $X, Y, Z, ...$ The atomic formulas are $t = u$, $X = Y$, $t \leq u$, $t \in X$ for any number terms t, u and string variables X, Y. Formulas are built from atomic formulas using $\wedge, \vee, \neg$ and both number and string quantifiers $\exists x, \exists X, \forall x, \forall X$. Bounded number quantifiers are defined as usual, and the bounded string quantifier $\exists X \leq t\ \varphi$ stands for $\exists X(|X| \leq t \wedge \varphi)$ and $\forall X \leq t\ \varphi$ stands for $\forall X(|X| \leq t \supset \varphi)$, where X does not occur in the term t.

$\mathbf{\Sigma}_0^B$ is the set of all $\mathcal{L}_A^2$-formulas in which all number quantifiers are bounded and with no string quantifiers. $\mathbf{\Sigma}_1^B$ (corresponding to *strict* $\Sigma_1^{1,b}$ in [Kra95]) formulas begin with zero or more bounded existential string quantifiers, followed by a $\mathbf{\Sigma}_0^B$ formula. These classes are extended to $\mathbf{\Sigma}_i^B$, $i \geq 2$, (and $\mathbf{\Pi}_i^B$, $i \geq 0$) in the usual way.

We use the notation $\mathbf{\Sigma}_0^B(\mathcal{L})$ to denote $\mathbf{\Sigma}_0^B$ formulas which may have two-sorted function and predicate symbols from the vocabulary $\mathcal{L}$ in addition to the basic vocabulary $\mathcal{L}_A^2$.

Two-sorted complexity classes contain relations $R(\vec{x}, \vec{X})$ (and possibly number-valued functions $f(\vec{x}, \vec{X})$ or string-valued functions $F(\vec{x}, \vec{X})$), where the arguments $\vec{x} = x_1, \ldots, x_k$ range over $\mathbb{N}$, and $\vec{X} = X_1, \ldots, X_\ell$ range over finite subsets of $\mathbb{N}$. In defining complexity classes using machines or circuits, the number arguments x_i are presented in unary notation (a string of x_i ones), and the arguments X_i are presented as bit strings. Thus the string arguments are the important inputs, and the number arguments are small auxiliary inputs useful for indexing the bits of strings.

As mentioned before, uniform $\mathbf{AC}^0$ has several equivalent characterizations [Imm99], including **LTH** (the log time hierarchy on alternating Turing machines) and **FO** (describable by a first-order formula using predicates for plus and times). Thus in the two-sorted setting we can define $\mathbf{AC}^0$ to be the class of relations $R(\vec{x},\vec{X})$ such that some alternating Turing machine accepts R in time $O(\log n)$ with a constant number of alternations, using the input conventions for numbers and strings given above. Then from the **FO** characterization of $\mathbf{AC}^0$ we obtain the following nice connection between $\mathbf{AC}^0$ and our two-sorted $\mathcal{L}_A^2$-formulas.

Theorem 9 ($\mathbf{\Sigma}_0^B$ Representation Theorem)**.** *A relation $R(\vec{x},\vec{X})$ is in $\mathbf{AC}^0$ iff it is represented by some $\mathbf{\Sigma}_0^B$ formula $\varphi(\vec{x},\vec{X})$.*

In general, if $\mathbf{C}$ is a class of relations (such as $\mathbf{AC}^0$) then we want to associate a class $\mathbf{FC}$ of functions with $\mathbf{C}$. Here $\mathbf{FC}$ will contain string-valued functions $F(\vec{x},\vec{X})$ and number-valued functions $f(\vec{x},\vec{X})$. We require that these functions be p-bounded; i.e. for each F and f there is a polynomial $g(n)$ such that $|F(\vec{x},\vec{X})| \leq g(max(\vec{x},|\vec{X}|))$ and $f(\vec{x},\vec{X}) \leq g(max(\vec{x},|\vec{X}|))$.

We define the *bit graph* $B_F(i,\vec{x},\vec{X})$ by

$$B_F(i,\vec{x},\vec{X}) \leftrightarrow F(\vec{x},\vec{X})(i) \tag{5}$$

Definition 10. *If $\mathbf{C}$ is a two-sorted complexity class of relations, then the corresponding function class $\mathbf{FC}$ consists of all p-bounded number functions whose graphs are in $\mathbf{C}$, together with all p-bounded string functions whose bit graphs are in $\mathbf{C}$.*

For example, binary addition $F_+(X,Y) = X + Y$ is in $\mathbf{FAC}^0$, but binary multiplication $F_\times(X,Y) = X \cdot Y$ is not.

Definition 11. *A string function is* $\mathbf{\Sigma}_0^B$-definable *from a collection $\mathcal{L}$ of two-sorted functions and relations if it is p-bounded and its bit graph is represented by a $\mathbf{\Sigma}_0^B(\mathcal{L})$ formula. Similarly, a number function is* $\mathbf{\Sigma}_0^B$-definable *from $\mathcal{L}$ if it is p-bounded and its graph is represented by a $\mathbf{\Sigma}_0^B(\mathcal{L})$ formula.*

It is not hard to see that $\mathbf{FAC}^0$ is closed under $\mathbf{\Sigma}_0^B$-definability, meaning that if the bit graph of F is represented by a $\mathbf{\Sigma}_0^B(\mathbf{FAC}^0)$ formula, then F is already in $\mathbf{FAC}^0$.

In order to define complexity classes such as $\mathbf{AC}^0(m)$ and $\mathbf{TC}^0$, as well as relativized classes such as $\mathbf{AC}^0(\alpha)$, we need to iterate $\mathbf{\Sigma}_0^B$-definability to obtain the notion of $\mathbf{AC}^0$ reduction.

Definition 12. *We say that a string function F (resp. a number function f) is $\mathbf{AC}^0$-reducible to $\mathcal{L}$ if there is a sequence of string functions $F_1,\ldots,F_n$ $(n \geq 0)$ such that*

$$F_i \textit{ is } \mathbf{\Sigma}_0^B\textit{-definable from } \mathcal{L} \cup \{F_1,\ldots,F_{i-1}\},\ \textit{for } i = 1,\ldots,n; \tag{6}$$

and F (resp. f) is $\mathbf{\Sigma}_0^B$-definable from $\mathcal{L} \cup \{F_1,\ldots,F_n\}$. A relation R is $\mathbf{AC}^0$-reducible to $\mathcal{L}$ if there is a sequence $F_1,\ldots,F_n$ as above, and R is represented by a $\mathbf{\Sigma}_0^B(\mathcal{L} \cup \{F_1,\ldots,F_n\})$ formula.

In other words, F is $\mathbf{AC}^0$-reducible to $\mathcal{L}$ if there is a uniform constant-depth polysize circuit family that computes F, where the circuits are allowed gates (each of depth one) which compute the functions and predicates in $\mathcal{L}$ (as well as the Boolean connectives).

For each class $\mathbf{C}$ in

$$\{\mathbf{TC}^0, \mathbf{AC}^0(m), \mathbf{NC}^1, \mathbf{L}, \mathbf{NL}\} \tag{7}$$

we consider a natural complete relation $R_{\mathbf{C}}$ as follows:

- $\mathbf{TC}^0$: $\mathrm{Numones}(k, X)$ holds iff k is the number of 1 bits in the binary string X.
- $\mathbf{AC}^0(m)$: $\mathrm{Mod}_m(X)$ holds iff the number of 1 bits in X is 1 modulo m.
- $\mathbf{NC}^1$: $\mathrm{Mfvp}(X)$ holds iff X codes a true balanced monotone Boolean sentence. ("Mfvp" stands for "monotone formula value problem")
- $\mathbf{L}$: $\mathrm{Spath}(s, t, G)$ holds iff G codes a directed graph with outdegree at most 1, and s, t are two vertices of G, and there is a path in G from s to t.
- $\mathbf{NL}$: $\mathrm{Conn}(s, t, G)$ holds iff G is a directed graph that contains a path from s to t.

The following result follows easily from the definitions of the complexity classes and well-known complete problems:

Theorem 13. *Each class $\mathbf{C}$ in* (7) *is the class of relations $\mathbf{AC}^0$-reducible to $R_{\mathbf{C}}$.*

Recall the relativized classes given in Definitions 1, 2, and 3.

Theorem 14. *For each class $\mathbf{C}(\alpha)$ in*

$$\{\mathbf{TC}^0(\alpha), \mathbf{AC}^0(m, \alpha), \mathbf{NC}^1(\alpha), cs\mathbf{L}(\alpha), cs\mathbf{NL}(\alpha)\}$$

$\mathbf{C}(\alpha)$ is the class of relations $\mathbf{AC}^0$-reducible to $\{R_{\mathbf{C}}, \alpha\}$.

Proof. For the classes $\mathbf{TC}^0(\alpha), \mathbf{AC}^0(m, \alpha), \mathbf{NC}^1(\alpha)$ this is immediate from the definitions involved. For the classes $\mathrm{cs}\mathbf{L}(\alpha), \mathrm{cs}\mathbf{NL}(\alpha)$ we show they are $\mathbf{AC}^0$-reducible to their corresponding path problem and α using ideas in the proof of Theorem 5. Conversely, to show that a relation that is $\mathbf{AC}^0$-reducible to the path problem and α is in the corresponding class $\mathrm{cs}\mathbf{L}(\alpha)$ or $\mathrm{cs}\mathbf{NL}(\alpha)$, the Turing machine performs a depth-first search of the constant-depth reducing circuit. Each α query is answered using the constant-height oracle stack, and each path query is answered by simulating the log-space Turing machine that solves that query, where each input bit of the query must be recomputed each time it is needed in the computation. □

The following corollary is immediate from the two preceding theorems and the transitivity of $\mathbf{AC}^0$-reducibility. It generalizes results in [Wil89].

Corollary 15. *For any $\mathbf{C}_1, \mathbf{C}_2$ in* (7) *$\mathbf{C}_1 = \mathbf{C}_2$ if and only if for all α, $\mathbf{C}_1(\alpha) = \mathbf{C}_2(\alpha)$, where $\mathbf{L}(\alpha)$ means $cs\mathbf{L}(\alpha)$ and $\mathbf{NL}(\alpha)$ means $cs\mathbf{NL}(\alpha)$.*

B1. $x + 1 \neq 0$	**B7.** $(x \leq y \wedge y \leq x) \supset x = y$
B2. $x + 1 = y + 1 \supset x = y$	**B8.** $x \leq x + y$
B3. $x + 0 = x$	**B9.** $0 \leq x$
B4. $x + (y + 1) = (x + y) + 1$	**B10.** $x \leq y \vee y \leq x$
B5. $x \cdot 0 = 0$	**B11.** $x \leq y \leftrightarrow x < y + 1$
B6. $x \cdot (y + 1) = (x \cdot y) + x$	**B12.** $x \neq 0 \supset \exists y \leq x(y + 1 = x)$
L1. $X(y) \supset y < \lvert X\rvert$	**L2.** $y + 1 = \lvert X\rvert \supset X(y)$
SE. $[\lvert X\rvert = \lvert Y\rvert \wedge \forall i < \lvert X\rvert(X(i) \leftrightarrow Y(i))] \;\supset\; X = Y$	

Figure 1: 2-**BASIC**

3.2 Nonrelativized Theories

The theory $\mathbf{V}^0$ (essentially $\mathbf{\Sigma}_0^p$-*comp* in [Zam96], and $\mathbf{I\Sigma}_0^{1,b}$ (without #) in [Kra95]) is the theory over $\mathcal{L}_A^2$ that is axiomatized by the axioms listed in Figure 1 together with the axiom scheme $\mathbf{\Sigma}_0^B(\mathcal{L}_A^2)$-**COMP**, i.e. the set of all formulas of the form

$$\exists X \leq y \forall z < y(X(z) \leftrightarrow \varphi(z)), \tag{8}$$

where $\varphi(z)$ is any formula in $\mathbf{\Sigma}_0^B(\mathcal{L}_A^2)$, and X does not occur free in $\varphi(z)$.

Using the the $\mathbf{\Sigma}_0^B$ Representation Theorem 9, it can be shown that a p-bounded function is in $\mathbf{FAC}^0$ if and only if it is provably total (i.e., $\mathbf{\Sigma}_1^B$ definable) in $\mathbf{V}^0$.

More generally, for various subclasses $\mathbf{C}$ of $\mathbf{P}$, a theory $\mathbf{VC}$ is developed in [CN06, Chapter 9] that characterizes $\mathbf{C}$ in the sense that the functions in $\mathbf{FC}$ are precisely the provably total functions of $\mathbf{VC}$. (The theory for $\mathbf{AC}^0(m)$ is $\mathbf{V}^0(m)$.) The theory $\mathbf{VC}$ is axiomatized by the axioms of $\mathbf{V}^0$ together with an axiom that formalizes a polytime computation of an answer for a complete problem of $\mathbf{C}$, asuming the parameters as given inputs. For a class $\mathbf{C}$ in (7), the complete problem is $R_{\mathbf{C}}$.

To formulate these axioms we introduce the pairing function $\langle y, z\rangle$, which stands for the term $(y + z)(y + z + 1) + 2z$. This allows us to interpret a string X as a two-dimensional bit array, using the notation

$$X(y, z) \equiv X(\langle y, z\rangle) \tag{9}$$

For example, a graph with a vertices can be encoded by a pair (a, E) where $E(u, v)$ holds iff there is an edge from u to v, for $0 \leq u, v < a$. The theory **VNL** is axiomatized by $\mathbf{V}^0$ and $CONN \equiv \forall a \forall E \exists Y\, \delta_{CONN}(a, E, Y)$. The formula $\delta_{CONN}(a, E, Y)$ states that for the graph encoded by (a, E), Y encodes a polytime computation of the nodes that are reachable from nodes 0: $Y(z, x)$ holds iff there is a path from 0 to x of length $\leq z$.

$$\begin{aligned}\delta_{CONN}(a, E, Y) \equiv\ & Y(0, 0) \wedge \forall x < a(x \neq 0 \supset \neg Y(0, x)) \wedge \\ & \forall z < a \forall x < a\, [Y(z + 1, x) \leftrightarrow (Y(z, x) \vee \exists y < a,\ Y(z, y) \wedge E(y, x))]\,.\end{aligned}$$

The additional axioms for other theories are listed below. Here $(Z)^x$ is the x-th element of the sequence of numbers encoded by Z:

$$y = (Z)^x \leftrightarrow (y < |Z| \wedge Z(x,y) \wedge \forall z < y \neg Z(x,z)) \vee \\ (\forall z < |Z| \neg Z(x,z) \wedge y = |Z|)$$

Also, $\log a$, or $|a|$, denotes the integral part of $\log_2(a)$. Note that this function is provably total in $\mathbf{V}^0$. The $\mathbf{\Sigma}_0^B$ formulas below contain the functions $(Z)^x$ and $|a|$, but these functions can be eliminated using their $\mathbf{\Sigma}_0^B$ defining axioms.

- $\mathbf{VTC}^0$: $\mathit{NUMONES} \equiv \forall X \forall x \exists Y \, \delta_{NUM}(x,X,Y)$ where

  $$\delta_{NUM}(x,X,Y) \equiv \ (Y)^0 = 0 \wedge \\ \forall z < x \left[(X(z) \supset (Y)^{z+1} = (Y)^z + 1) \wedge (\neg X(z) \supset (Y)^{z+1} = (Y)^z) \right]$$

  (For $z \geq 1$, $(Y)^z$ is the number of 1 bits in $X(0), X(1), \ldots, X(z-1)$.)

- $\mathbf{V}^0(m)$ (the theory for $\mathbf{AC}^0(m)$): $\mathbf{MOD}_m \equiv \forall X \forall x \exists Y \, \delta_{\mathbf{MOD}_m}(x,X,Y)$ where

  $$\delta_{\mathbf{MOD}_m}(x,X,Y) \equiv Y(0,0) \wedge \\ \forall z < x \left[(X(z) \supset (Y)^{z+1} = ((Y)^z + 1) \mod m)) \wedge (\neg X(z) \supset (Y)^{z+1} = (Y)^z) \right].$$

  (For $z \geq 1$, $(Y)^z$ is the number of 1 bits in $X(0), X(1), \ldots, X(z-1)$ modulo m.)

- $\mathbf{VNC}^1$: $\mathit{MFVP} \equiv \forall a \forall G \forall I \exists Y \, \delta_{MFVP}(a,G,I,Y)$ where

  $$\delta_{MFVP}(a,G,I,Y) \equiv \ \forall x < a \, [Y(x+a) \leftrightarrow I(x) \wedge \\ 0 < x \supset [Y(x) \leftrightarrow [(G(x) \wedge Y(2x) \wedge Y(2x+1)) \vee (\neg G(x) \wedge (Y(2x) \vee Y(2x+1)))]]]$$

  (For the formula viewed as a balanced binary tree encoded by (a,G)— node x's children are $2x$ and $2x+1$, and $G(x)$ indicates whether node x is an $\vee$ or $\wedge$ node— $Y(x)$ is the value of node x when the inputs are given by I.)

- $\mathbf{VL}$: $\mathit{SinglePATH}$ is the axiom

  $$[\forall x < a \exists! y < a E(x,y)] \ \supset \ \exists P \ \left[(P)^0 = 0 \wedge \forall v < a E((P)^v, (P)^{v+1}) \right]$$

  ($(P)^v$ is the vertex of distance v from 0.)

- $\mathbf{VAC}^k$: $\forall a \forall E \forall G \forall I \exists Y \, \delta_{MCVP}(a, |a|^k, E, G, I, Y)$ where

  $$\delta_{MCVP}(w,d,E,G,I,Y) \equiv \forall x < w \forall z < d \, [(Y(0,x) \leftrightarrow I(x)) \wedge \\ (Y(z+1,x) \leftrightarrow [[G(z+1,x) \wedge \forall u < w \, (E(z,u,x) \supset Y(z,u))] \vee \\ [\neg G(z+1,x) \wedge \exists u < w \, (E(z,u,x) \wedge Y(z,u))]])]$$

("MCVP" stands for "monotone circuit value problem". Here the formula $\delta_{MCVP}(w,d,E,G,I,Y)$ states that given input I to a circuit encoded by (w,d,E,G)—there are w gates on each of the the d layers, the gate connection is given by E and the gates are specified by G—Y encodes an evaluation of the gates.)

- $\mathbf{VNC}^k$ (for $k \geq 2$):

$$\forall a \forall E \forall G \forall I (\mathit{Fanin2}(a, |a|^k, E) \supset \exists Y\, \delta_{MCVP}(a, |a|^k, E, G, I, Y))$$

  Here $\mathit{Fanin2}(w,d,E)$ states that the gates have fanin at most 2:

$$\forall z < d \forall x < w \exists u_1, u_2 < w \forall v < w\, (E(z,v,x) \supset v = u_1 \vee v = u_2)$$

Showing that the functions in $\mathbf{FC}$ are precisely the provably total functions of $\mathbf{VC}$ can be done by first developing an universal theory $\overline{\mathbf{VC}}$ whose underlying vocabulary consists of all functions in $\mathbf{FC}$ with their defining axioms. The provably total functions of $\overline{\mathbf{VC}}$ are precisely the functions in $\mathbf{FC}$, so it remains to show that $\overline{\mathbf{VC}}$ is a conservative extension of $\mathbf{VC}$ [CN06, Corollary 9.33].

Our goal for the remainder of this section is to obtain relativized theories $\mathbf{VC}(\alpha)$ that characterize the relativized classes discussed in Section 2. We will use the results of [CN06, Chapter 9] and the fact that the axioms in $\mathbf{VC}(\alpha)$ encode the polytime computation of corresponding $\mathbf{AC}^0$-complete problems of the classes (cf. Theorem 14).

3.3 Relativized Theories

First note that a sequence of strings can be encoded using the string function Row, where

$$\mathit{Row}(x,Z)(i) \leftrightarrow i < |Z| \wedge Z(x,i)$$

($\mathit{Row}(x,Z)$ will be also written as $Z^{[x]}$.)

Notation For a predicate α, let $\mathbf{\Sigma}_0^B(\alpha)$ denote the class of $\mathbf{\Sigma}_0^B$ formulas in $\mathcal{L}_A^2 \cup \{\mathit{Row}, \alpha\}$.

Definition 16. $\mathbf{V}^0(\alpha) = \mathbf{V}^0 + \mathbf{\Sigma}_0^B(\alpha)$-$\mathbf{COMP}$. *For each class* $\mathbf{C}$ *in* (7)*, the theory* $\mathbf{VC}(\alpha)$ *is defined as* $\mathbf{VC}$ *with* $\mathbf{\Sigma}_0^B$-$\mathbf{COMP}$ *replaced by* $\mathbf{\Sigma}_0^B(\alpha)$-$\mathbf{COMP}$.

Notice that natural relativized versions of the additional axioms of $\mathbf{VC}$, such as CONN, are already provable in $\mathbf{VC}(\alpha)$. For example, let $\mathit{CONN}(\alpha)$ be the axiom scheme

$$\begin{aligned}\forall a \exists Y\ [&Y(0,0) \wedge \forall x < a (x \neq 0 \supset \neg Y(0,x)) \wedge \\ &\forall z < a \forall x < a,\ Y(z+1,x) \leftrightarrow (Y(z,x) \vee \exists y < a,\ Y(z,y) \wedge \varphi(y,x))].\end{aligned}$$

where φ is a $\mathbf{\Sigma}_0^B(\alpha)$ formula. Then it is easy to use $\mathbf{\Sigma}_0^B(\alpha)$-$\mathbf{COMP}$ to show that $\mathbf{VNL}(\alpha) \vdash \mathit{CONN}(\alpha)$.

Theorem 17. *For a class* $\mathbf{C}$ *in* $\{\mathbf{AC}^0, \mathbf{AC}^0(m), \mathbf{TC}^0, \mathbf{NC}^1, \mathbf{L}, \mathbf{NL}\}$*, a function is in* $\mathbf{FC}(\alpha)$ *if and only if it is provably total in* $\mathbf{VC}(\alpha)$*.*

Proof. Consider the ($\Longrightarrow$) direction for $\mathbf{C} = \mathbf{AC}^0$. Here $\mathbf{FAC}^0(\alpha)$ consists of all p-bounded functions which are $\mathbf{AC}^0$-reducible to $\{\alpha\}$. The fact that $\mathbf{V}^0(\alpha)$ can define all functions in $\mathbf{FAC}^0(\alpha)$ can be proved by induction on n in Definition 12.

For each other class, the ($\Longrightarrow$) direction can also be proved by induction on n from Definition 12 using Theorem 14.

The ($\Longleftarrow$) direction can be proved by a standard witnessing argument, i.e., by induction on the length of a free-cut-free $\mathbf{VC}(\alpha)$-proof whose end-sequent is the defining axiom of a function provably total in $\mathbf{VC}(\alpha)$. $\square$

Now we present the theories $\mathbf{VAC}^k(\alpha)$ (for $k \geq 1$) and $\mathbf{VNC}^k(\alpha)$ (for $k \geq 2$). Note that the problem of evaluating uniform $\mathbf{AC}^k(\alpha)$ (or $\mathbf{NC}^k(\alpha)$) circuits is $\mathbf{AC}^0$-complete for the corresponding relativized class. Thus $\mathbf{VAC}^k(\alpha)$ (or $\mathbf{VNC}^k(\alpha)$) will be axiomatized by $\mathbf{V}^0$ together with an additional axiom that formalizes a polytime computation that solves the respective complete problem.

First we formalize a polytime evaluation of an oracle circuit $C = (w, d, E, G)$ given input I. Since the order of inputs to an oracle gate is important, the edge relations of the underlying graph is now encoded by a string variable E, where $E(z, t, u, x)$ indicates that gate u on layer z is the t-th input to gate x on layer $z + 1$. The condition we need for E is

$$Proper(w, d, E) \equiv \forall z < d \forall t, x, u_1, u_2 < w,\ (E(z, t, u_1, x) \wedge E(z, t, u_2, x)) \supset u_1 = u_2$$

In the formula $\delta^{\alpha}_{MCVP}(w, d, E, G, I, Q, Y)$ defined below, $Q^{[z+1,x]}$ encodes the query to the oracle gate x on layer $z + 1$. Here the type of gate x on layer z is specified by $(G)^{\langle z,x\rangle}$.

Definition 18. *The formula* $\delta^{\alpha}_{MCVP}(w, d, E, G, I, Q, Y)$ *is the formula*

$$\begin{aligned}
&\forall z < d \forall x < w \\
&[\forall t < w(Q^{[z+1,x]}(t) \leftrightarrow (\exists u < w,\ E(z, t, u, x) \wedge Y(z, u)))] \ \wedge\ [Y(0, x) \leftrightarrow I(x)] \wedge \\
&\quad [Y(z + 1, x) \leftrightarrow (((G)^{\langle z+1,x\rangle} = \text{“}\wedge\text{”} \wedge \forall t, u < w,\ E(z, t, u, x) \supset Y(z, u)) \vee \\
&\qquad ((G)^{\langle z+1,x\rangle} = \text{“}\vee\text{”} \wedge \exists t, u < w,\ E(z, t, u, x) \wedge Y(z, u)) \vee \\
&\qquad\qquad ((G)^{\langle z+1,x\rangle} = \text{“}\alpha\text{”} \wedge \alpha(Q^{[z+1,x]}))))]
\end{aligned}$$

Definition 19 ($\mathbf{VAC}^k(\alpha)$)**.** *For* $k \geq 1$*,* $\mathbf{VAC}^k(\alpha)$ *is the theory over the vocabulary* $\mathcal{L}^2_A \cup \{Row, \alpha\}$ *and is axiomatized by the axioms of* $\mathbf{V}^0$ *and the following axiom:*

$$\forall w, E, G, I(Proper(w, d, E) \supset \exists Q, Y \delta^{\alpha}_{MCVP}(w, (\log w)^k, E, G, I, Q, Y))$$

To specify an $\mathbf{NC}^k(\alpha)$ circuit, we need to express the condition that $\wedge$ and $\vee$ gates have fanin 2. Here we use the following formula $Fanin2'(w, d, E, G)$:

$$\forall z < d \forall x < w((G)^{\langle z,x\rangle} \neq \text{“}\alpha\text{”} \supset \exists u_1, u_2 < w \forall t, v < w,\ E(z, t, v, x) \supset v = u_1 \vee v = u_2)$$

Moreover, the nested depth of oracle gates in circuit (w, d, E, G) needs to be bounded. The formula $OHeight(w, d, h, E, G, H)$ below states that this nested depth is bounded by h:

$$(\forall z \leq d \forall x < w \exists! s \leq h\, H(z, x, s)) \wedge (\forall x < w H(0, x, 0)) \wedge$$
$$\forall z < d \forall x < w \exists m [m = max\{\ell : \exists t, u < w E(z, t, u, x) \wedge H(z, u, \ell)\} \wedge$$
$$[((G)^{\langle z+1, x \rangle} = \text{“}\alpha\text{”} \supset H(z+1, x, m+1)) \wedge ((G)^{\langle z+1, x \rangle} \neq \text{“}\alpha\text{”} \supset H(z+1, x, m))]]$$

Definition 20 **($\mathbf{VNC}^k(\alpha)$).** *For $k \geq 2$, $\mathbf{VNC}^k(\alpha)$ is the theory over $\mathcal{L}_A^2 \cup \{Row, \alpha\}$ and is axiomatized by $\mathbf{V}^0$ and the axiom*

$$\forall w \forall E, G, I, H,\ [Proper(w, d, E) \wedge Fanin2'(w, |w|^k, E, G) \wedge$$
$$OHeight(w, d, |w|^{k-1}, E, G, H)] \ \supset\ \exists Q, Y \delta^{\alpha}_{MCVP}(w, (\log w)^k, E, G, I, Q, Y)$$

The next theorem can be proved in the same way as Theorem 17.

Theorem 21. *For $k \geq 1$, the functions in $\mathbf{FAC}^k(\alpha)$ are precisely the provably total functions of $\mathbf{VAC}^k(\alpha)$. The same holds for $\mathbf{FNC}^k(\alpha)$ and $\mathbf{VNC}^k(\alpha)$, for $k \geq 2$.*

4 Separation Results

One of the obvious benefits of considering relativized complexity classes is that separations are at hand. Even though the unrelativized inclusion $\mathbf{AC}^1 \subseteq \mathbf{PH}$ is strongly conjectured to be strict, no proof is currently known. On the other hand, relative to an oracle the $\mathbf{AC}^k$-hierarchy is strict. Here we reconstruct a technique used by Takeuti [Tak95] to separate theories in weak bounded arithmetic in a circuit-theoretic setting. Using the hierarchy result together with the witnessing theorem we obtain an unconditional separation of our relativized theories.

The idea is that computing the k'th iterate $f^k(0) = f(f(\ldots f(0)))$ of a function f is essentially a sequential procedure, whereas shallow circuits represent parallel computation. So a circuit performing well in a sequential task has to be deep. To avoid that the sequential character of the problem can be circumvented by precomputing all possible values, the domain of f is chosen big enough; we will consider functions $f\colon [2^n] \to [2^n]$.

Of course with such a big domain, we cannot represent such functions simply by a value table. That's how oracles come into play: oracles allow us to provide a predicate on strings as input, without the need of having an input bit for every string. In fact, the number of bits potentially accessible by an oracle gate is exponential in the number of its input wires.

Therefore we represent the i'th bit of $f(x)$ for $x \in \{0,1\}^n$ by whether or not the string $x\underline{i}$ belongs to the language of the oracle. Here $\underline{i}$ is some canonical coding of the natural number i using $\log(n)$ bits.

Our argument can be summarized as follows. We assume a circuit of height h be given that supposedly computes the ℓ'th iterate of any function f given by the oracle. Then we construct, step by step, an oracle that fools this circuit, if $\ell > h$. To do so, for each layer of the circuit we decide how to answer the oracle questions, and we do this in a way that is consistent with the previous layers and such that all the circuit at layer i knows about f is at most the value of $f^i(0)$. Of course, to make this step-by-step construction possible we have to consider partial functions during our construction.

If A and B are sets we denote by $f\colon A \rightharpoonup B$ that f is a partial function from A to B. In other words, f is a function, its domain $\mathrm{dom}(f)$ is a subset of A and its range $\mathrm{rng}(f)$ is a subset of B.

Definition 22. A partial function $f\colon [2^n] \rightharpoonup [2^n]$ is called *ℓ-sequential* if for some $k \leq \ell$ it is the case that $0, f(0), f^2(0), \ldots, f^k(0)$ are all defined, but $f^k(0) \notin \mathrm{dom}(f)$.

Note that in Definition 22 it is necessarily the case that $0, f(0), f^2(0), \ldots, f^k(0)$ are distinct.

Lemma 23. *Let $n \in \mathbb{N}$ and $f\colon [2^n] \rightharpoonup [2^n]$ be an ℓ-sequential partial function. Moreover, let $M \subset [2^n]$ such that $|\mathrm{dom}(f) \cup M| < 2^n$. Then there is a $(\ell+1)$-sequential extension $f' \supseteq f$ with $\mathrm{dom}(f') = \mathrm{dom}(f) \cup M$.*

Proof. Let $a \in [2^n] \setminus (M \cup \mathrm{dom}(f))$. Such an a exists by our assumption on the cardinality of $M \cup \mathrm{dom}(f)$. Let f' be f extended by setting $f'(x) = a$ for all $x \in M \setminus \mathrm{dom}(f)$. This f' is as desired.

Indeed, assume that $0, f'(0), \ldots, f'^{\ell+1}(0), f'^{\ell+2}(0)$ are all defined. Then, since $a \notin \mathrm{dom}(f')$, all the $0, f'(0), \ldots, f'^{\ell+1}(0)$ have to be different from a. Hence these values have already been defined in f. But this contradicts the assumption that f was ℓ-sequential. $\square$

Definition 24. To any natural number n and any partial function $f\colon [2^n] \rightharpoonup [2^n]$ we associate a its *bit graph* $\alpha_{n,f}$ as a partial function $\alpha_{n,f}\colon \{0,1\}^{n+\log n} \rightharpoonup \{0,1\}$ in the obvious way. More precisely, $\alpha_{n,f}(uv)$ is the i'th bit of $f(x)$ if $f(x)$ is defined, and undefined otherwise, where u is a string of length n coding the natural number x and v is a string of length $\log n$ coding the natural number i.

If $f\colon [2^n] \to [2^n]$ is a total function, we define the set $\mathrm{A}_f = \{x \mid \alpha_{n,f}(x) = 1\} \subseteq \{0,1\}^{n+\log n}$.

Immediately from Definition 24 we note that f can be uniquely reconstructed from A_f. If $A \subseteq \{0,1\}^*$ is a set of bitstrings, we denote by $A^{[n]} = \{x \in A \mid |x| = n + \log n\}$ the set of all strings in A of length $n + \log n$.

In what follows, circuits refer to oracle circuits as discussed in Section 2.1. We are mainly interested in circuits with no Boolean inputs, so the output depends only on the oracle.

Theorem 25. *Let C be any circuit of depth h and size strictly less then 2^n. If C on oracle A computes correctly $f^\ell(0)$ for the (uniquely determined) $f\colon [2^n] \to [2^n]$ such that $\mathrm{A}_f = A^{[n]}$, and this is true for all oracles A, then $\ell \leq h$.*

Proof. Assume that such a circuit computes $f^\ell(0)$ correctly for all oracles. We have to find an oracle that witnesses $\ell \le h$. First fix the oracle arbitrarily on all strings of length different from $n + \log n$. So, in effect we can assume that the circuit only uses oracle gates with $n + \log n$ inputs.

By induction on $k \ge 0$ we define partial functions $f_k \colon [2^n] \rightharpoonup [2^n]$ with the following properties. (Here we number the *levels* of the circuit $0, 1, \dots, h-1$.)

- $f_0 \subseteq f_1 \subseteq f_2 \subseteq \dots$
- The size $|\mathrm{dom}(f_k)|$ of the domain of f_k is at most the number of oracle gates in levels strictly smaller than k.
- α_{n,f_k} determines the values of all oracle gates at levels strictly smaller than k.
- f_k is k-sequential.

We can take f_0 to be the totally undefined function, since $f^0(0) = 0$ by definition. As for the induction step let M be the set of all x of length n such that, for some $i < n$, the string $x\underline{i}$ is queried by an oracle gate at level k and let f_{k+1} be a k+1-sequential extension of f_k to domain $\mathrm{dom}(f_k) \cup M$ according to Lemma 23.

For $k = h$ we get the desired bound. As α_{n,f_h} already determines the values of all gates, the output of the circuit is already determined, but $f^{h+1}(0)$ is still undefined and we can define it in such a way that it differs from the output of the circuit. □

Inspecting the proof of Theorem 25 we note that it does not at all use what precisely the non-oracle gates compute, as long as the value only depends on the input, not on the oracle. In particular, the proof still holds if we consider subcircuits without oracle gates as a single complicated gate. Thus we have the following corollary of Theorem 5.

Corollary 26. *cs*$\mathbf{NL}(\alpha)$ *can iterate a function given by an oracle only constantly far. In particular, cs*$\mathbf{NL}(\alpha)$ *is a strict subclass of* $\mathbf{AC}^1(\alpha)$*.*

Having obtained a lower bound on the depth of an individual circuit, it is a routine argument to separate the corresponding circuit classes. In other words, we are now interested in finding one oracle that simultaneously witnesses that the $\mathbf{AC}^k(\alpha)$-hierarchy is strict. For the uniform classes this is possible by a simple diagonalization argument; in fact, the only property of uniformity we need is that there are at most countably many members in each complexity class. So we will use this as the definition of uniformity. It should be noted that this includes all the known uniformity notions.

Definition 27. If $g \colon \mathbb{N} \to \mathbb{N}$ is a function from the natural numbers to the natural numbers, and $A \subseteq \{0,1\}^*$ an oracle, we define the language

$$\begin{aligned}\mathcal{L}_g^A = \{x \mid\ & \text{the last bit of } f^{g(n)}(0) \text{ is 1,} \\ & \text{where } n = |x| \text{ and } f \text{ is such that } A^{[n]} = \mathrm{A}_f\}\end{aligned}$$

We note that in Definition 27 the f is uniquely determined by A and the length of x. Also, for logspace-constructible g the language $\mathcal{L}_g^A$ can be computed by logspace-uniform circuits of depth $g(n)$ and size $n \cdot g(n)$.

Recall that a circuit family is a sequence $(C_n)_{n\in\mathbb{N}}$ of circuits, such that C_n has n inputs and one output. The language of a circuit family $(C_n)_{n\in\mathbb{N}}$ is the set of all strings $x \in \{0,1\}^*$ such that the output of $C_{|x|}$ with input x is 1.

Definition 28. A *notion of uniformity* is any countable set $\mathcal{U}$ of circuit families.

Let $\mathcal{U}$ be a notion of uniformity, and $h, s \colon \mathbb{N} \to \mathbb{N}$ functions. The $\mathcal{U}$-uniform h, s-circuits are those circuit families $(C_n)_{n\in\mathbb{N}} \in \mathcal{U}$ of $\mathcal{U}$ such that C_n has depth at most $h(n)$ and size at most $s(n)$.

By a simple diagonalization argument we obtain the following theorem.

Theorem 29. *Let $\mathcal{U}$ be a notion of uniformity and h_c a family of functions such that for all $c \in \mathbb{N}$ the function h_{c+1} eventually strictly dominates h_c. Moreover, let s_c be a family of strictly subexponentially growing functions. Then there is a single oracle $A \subseteq \{0,1\}^*$ that simultaneously witnesses that $\mathcal{L}_{h_{c+1}}^A$ cannot be computed by $\mathcal{U}$-uniform h_c, s_c-circuits.*

Corollary 30. *There is a single oracle $A \subseteq \{0,1\}^*$ for which the relativized versions of* AC^k *form a strict hierarchy.*

Corollary 31. *The theories* $\mathbf{VAC}^k(\alpha)$ *form a strict hierarchy.*

References

[BIS90] David A. Mix Barrington, Neil Immerman, and Howard Straubing. On Uniformity within NC^1. *Journal of Computer and System Sciences*, 41:274–306, 1990.

[Bus86] Jonathan Buss. Relativized Alternation. In *Proceedings, Structure in Complexity Theory Conference*. Springer-Verlag, 1986.

[CN06] Stephen Cook and Phuong Nguyen. Foundations of Proof Complexity: Bounded Arithmetic and Propositional Translations. Book in progress, 2006.

[Coo85] Stephen Cook. A Taxonomy of Problems with Fast Parallel Algorithms. *Information and Control*, 64(1-3):2–21, 1985.

[Imm99] Neil Immerman. *Descriptive Complexity*. Springer, 1999.

[Kra95] Jan Krajíček. *Bounded Arithmetic, Propositional Logic, and Complexity Theory*. Cambridge University Press, 1995.

[LL76] Richard Ladner and Nancy Lynch. Relativization of questions about log space computability. *Mathematical Systems Theory*, 10:19–32, 1976.

[NC05] Phuong Nguyen and Stephen Cook. Theory for $\mathbf{TC}^0$ and Other Small Complexity Classes. *Logical Methods in Computer Science*, 2, 2005.

[Orp83] P. Orponen. General Nonrelativizability Results for Parallel Models of Computation. In *Proceedings, Winter School in Theoretical Computer Science*, pages 194–205. 1983.

[RST84] Walter Ruzzo, Janos Simon, and Martin Tompa. Space-Bounded Hierarchies and Probabilistic Computations. *Journal of Computer and System Sciences*, 28(2):216–230, 1984.

[Sim77] Istvan Simon. *On some subrecursive reducibilities.* PhD thesis, Stanford University, 1977.

[Tak95] Gaisi Takeuti. Separations of Theories in Weak Bounded Arithmetic. *Annals of Pure and Applied Logic*, 71:47–67, 1995.

[Wil88] Christopher Wilson. A Measure of Relativized Space Which Is Faithful with Respect to Depth. *Journal of Computer and System Sciences*, 36:303–312, 1988.

[Wil89] Christopher Wilson. Relativized NC. *Mathematical Systems Theory*, 20:13–29, 1989.

[Zam96] Domenico Zambella. Notes on Polynomially Bounded Arithmetic. *Journal of Symbolic Logic*, 61(3):942–966, 1996.