⌐computational complexity

RELATIVIZING SMALL COMPLEXITY CLASSES AND THEIR THEORIES

Klaus Aehlig, Stephen Cook, and Phuong Nguyen

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 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. Finally, 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, and hence, the oracle separations imply separations for the relativized theories.

Keywords. Complexity classes, relativized classes, proof complexity, logical theories.

Subject classification. 68Q15.

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 **P** and **NP** by methods that relativize. However, similar results have not been established for subclasses of **P** such as **L** and **NL**. Indeed, prior to this work, there has not been a satisfying definition of the relativized version of **NL** that preserves simultaneously the inclusions.

(1.1) $$\mathbf{NC}^1 \subseteq \mathbf{L} \subseteq \mathbf{NL} \subseteq \mathbf{AC}^1.$$

(In this paper, $\mathbf{NC}^k$ and $\mathbf{AC}^k$ refer to their uniform versions.) For example (Ladner & Lynch 1976), 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 Ruzzo *et al.* (1984) that the nondeterministic oracle machines be deterministic whenever the oracle tape (or oracle stack) is nonempty. Then the inclusion $\mathbf{NL}(\alpha) \subseteq \mathbf{P}(\alpha)$ relativizes, but not all inclusions in (1.1).

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 (Buss 1986; Orponen 1983; Wilson 1988). For the model defined by Wilson (1988), 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 (1985), 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 (1988) are required to satisfy the condition that at any time,

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

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 $\mathbf{NL}(\alpha) \subseteq \mathbf{AC}^2(\alpha)$ (Wilson 1988); 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, it can then be shown that $\mathbf{NL}(\alpha)$ is contained in the $\mathbf{AC}^0(\alpha)$ closure of the reachability problem for directed graphs, while $\mathbf{L}(\alpha)$ equals 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 (1.1).

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 nonrelativized classes were equal, their complete problems would be equivalent under $\mathbf{AC}^0$ reductions, hence even more under $\mathbf{AC}^0(\alpha)$ reductions and therefore the relativized classes would coincide as well.) Separating the relativized classes is as hard as separating their nonrelativized counterparts. This nicely generalizes known results (Simon 1977; Wilson 1987, 1988).

On the other hand, oracles that separate the classes $\mathbf{AC}^k$ (for $k = 0, 1, 2, \ldots$) and $\mathbf{P}$ have been constructed (Wilson 1987). 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

$$(1.2) \qquad\qquad f(f(\ldots f(0) \ldots))$$

for all oracle function f. (Clearly (1.2) 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

$$(1.3) \qquad \mathbf{NL}(\alpha) \subsetneq \mathbf{AC}^1(\alpha) \subsetneq \mathbf{AC}^2(\alpha) \subsetneq \cdots \subsetneq \mathbf{P}(\alpha).$$

The idea of using (1.2) to separate relativized circuit classes is already present in the work of Takeuti (1995) 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 (Nguyen & Cook 2006, 2010), 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. An oracle that separates classes in (1.3) is shown in Section 3. In Section 4, we define the associated theories and show their separation using the oracle defined in Section 3.

2. 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 input bits), 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 (Barrington *et al.* 1990; Immerman 1999). 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.

Here an $\mathbf{AC}^0$ reduction refers to a 'Turing' style reduction. Thus, a problem A is $\mathbf{AC}^0$ reducible to a problem B if there is a uniform polynomial size constant-depth family of circuits computing A, where the circuits are allowed to have oracle gates for B, as well as Boolean gates.

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 2.1 ($\mathbf{AC}^k(\alpha)$, $\mathbf{AC}^0(m, \alpha)$, $\mathbf{TC}^0(\alpha)$). *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 fanin 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 Cook (1985), 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 nested depth 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 $\mathcal{O}((\log n)^{k-1})$. This restriction implies constant oracle nesting depth for $\mathbf{NC}^1(\alpha)$ circuits and allows us to

show that in the relativized world $\mathbf{NC}^1$ is still contained in $\mathbf{L}$. Also the circuit value problem (for oracle $\mathbf{NC}^k$ circuits) is still complete for $\mathbf{NC}^k$, as expected.

DEFINITION 2.2 ($\mathbf{NC}^k(\alpha)$). *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:

$$\mathbf{AC}^0(\alpha) \subseteq \mathbf{AC}^0(m, \alpha) \subseteq \mathbf{TC}^0(\alpha) \subseteq \mathbf{NC}^1(\alpha) \subseteq \mathbf{AC}^1(\alpha)$$
$$\subseteq \cdots \subseteq \mathbf{P}(\alpha)$$

Further the $\mathbf{AC}^0$-complete problems for $\mathbf{AC}^0(m)$, $\mathbf{TC}^0$ and $\mathbf{NC}^1$ are also $\mathbf{AC}^0(\alpha)$-complete for the corresponding relativized classes. This is expressed by the next result, using the following complete problems: MODm and THRESH (the threshold function) are $\mathbf{AC}^0$-complete for $\mathbf{AC}^0(m)$ and $\mathbf{TC}^0$ respectively, and FORMVAL (the Boolean formula value problem) is both $\mathbf{AC}^0$-complete and $\mathbf{AC}^0$-many-one complete for $\mathbf{NC}^1$.

PROPOSITION 2.3.

$$\text{(2.4)} \qquad\qquad \mathbf{AC}^0(m, \alpha) = \mathbf{AC}^0(\mathsf{MOD}m, \alpha)$$
$$\text{(2.5)} \qquad\qquad \mathbf{TC}^0(\alpha) = \mathbf{AC}^0(\mathsf{THRESH}, \alpha)$$
$$\text{(2.6)} \qquad\qquad \mathbf{NC}^1(\alpha) = \mathbf{AC}^0(\mathsf{FORMVAL}, \alpha)$$

PROOF.   Each class on the right is included in the corresponding class on the left because a query to the complete problem can be replaced by a circuit computing the query. Each class on the left is a subset of the corresponding class on the right because the queries to α on the left have bounded nesting depth.   □

Note that there is no similar characterization of $\mathbf{AC}^1(\alpha)$ or $\mathbf{P}(\alpha)$, because here the queries to α can have unbounded nesting depth.

2.2. Relativized logspace classes. To define oracle logspace classes, we use a modification of Wilson's stack model (Wilson 1988). An advantage is that the relativized classes defined here are closed under $\mathbf{AC}^0$ reductions. This is not true for the nonstack 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 (Ruzzo *et al.* 1984) that the machine is deterministic when the oracle stack is non empty or when it is in a push state.

DEFINITION 2.7 ($cs\mathbf{L}(\alpha)$, $cs\mathbf{NL}(\alpha)$). *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 2.8. $\mathbf{NC}^1(\alpha) \subseteq cs\mathbf{L}(\alpha) \subseteq cs\mathbf{NL}(\alpha) \subseteq \mathbf{AC}^1(\alpha)$.

PROOF. The second inclusion is immediate from the definitions, and the first can be proved as in the standard proof of the fact that $\mathbf{NC}^1 \subseteq \mathbf{L}$ (see also Wilson 1988). The last inclusion can actually be strengthened, as shown in the next theorem. □

The next theorem partly extends Proposition 2.3 to the two new classes. 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. We define **1-STCONN** to be the same, except we require that every node in G has out degree at most one. Then **STCONN** and **1-STCONN** are $\mathbf{AC}^0$-many-one complete for **NL** and **L**, respectively.

A $\text{cs}\mathbf{L}(\alpha)$ function is defined by allowing the $\text{cs}\mathbf{L}(\alpha)$ machine to write on a write-only output tape. Then the notion of many-one $\text{cs}\mathbf{L}(\alpha)$ reducibility is defined as usual.

THEOREM 2.9. **(i)** $\text{cs}\mathbf{L}(\alpha) = \mathbf{AC}^0(\text{1-STCONN}, \alpha)$

(ii) $\text{cs}\mathbf{NL}(\alpha) \subseteq \mathbf{AC}^0(\text{STCONN}, \alpha)$

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

PROOF.   We start by proving the inclusion $\mathbf{AC}^0(\text{1-STCONN}, \alpha) \subseteq \text{cs}\mathbf{L}(\alpha)$ in (i). A problem in $\mathbf{AC}^0(\text{1-STCONN}, \alpha)$ is given by a uniform polynomial size constant-depth circuit family $\{C_n\}_{n\in\mathbb{N}}$ with oracle queries to **1-STCONN** and α. A $\text{cs}\mathbf{L}(\alpha)$ machine M on an input x of length n performs a depth-first search of the circuit C_n with input x. Each α oracle gate at depth k is answered using an oracle query at stack height k, and each oracle query to **1-STCONN** is answered by a logspace computation.

Note that this argument does not work for the corresponding inclusion in (ii), because once the oracle stack is nonempty, a $\text{cs}\mathbf{NL}(\alpha)$ machine becomes deterministic and cannot answer oracle queries to **STCONN** (assuming $\mathbf{L} \neq \mathbf{NL}$).

Now we prove the inclusion (ii). (The corresponding inclusion in the equation (i) is proved similarly.) 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:

$$(2.10) \qquad u_0 = START, \ u_1 = ACCEPT, \ u_2, \ldots, u_{p(n)-1}.$$

Here a configuration u_i encodes information about the internal state, the content and head position of the work tape, and the

position of the input tape head, *but no explicit information about the oracle stack* (although the internal state might encode implicit information).

Given an input x of length n, we construct a sequence $G_0, \ldots, G_h$ of directed graphs such that the set V_k of nodes in G_k consists of all pairs (k, u), where u is a configuration and k is the current height of the oracle stack (so $0 \le k \le h$). Thus (k, u) represents a 'height k configuration.' Note that the computation of M on input x can be described by a sequence of nodes in $\bigcup_k V_k$. We want to define the edge set E_k so that

(2.11) $((k, u), (k, u')) \in E_k$ iff (k, u') can be the next height k configuration after (k, u).

The edges E_k in G_k comprise the union

$$E_k = E_k^0 \cup E_k^1.$$

Define E_k^0 to consist of all pairs $((k, u), (k, u'))$ such that u does not cause a push or pop and u' is a possible successor to u. If $k \ge 1$, then we also require that u be a deterministic configuration. Define E_k^1 to be empty if $k = h$, and if $0 \le k < h$, then E_k^1 consists of all pairs $((k, u), (k, u'))$ such that u is a deterministic configuration causing a push, and there is a sequence

(2.12) $(k + 1, v_0), \ldots, (k + 1, v_t)$

of configurations such that v_0 is the successor of u and $((k + 1, v_i), (k + 1, v_{i+1})) \in E_{k+1}$ for $0 \le i < t$ and v_t causes a pop and u' is the successor of v_t (given that (k, u) is the most recent level k node preceding $(k + 1, v_t)$).

Note that in a computation from (k, u) to (k, u') the sequence (2.12) is the sequence of height $k + 1$ nodes, and this sequence determines the string Q that is written on the height $k + 1$ oracle tape, and hence determines whether u' is the successor of v_t.

It is easy to prove (2.11) by induction on $k = h, h-1, \ldots, 0$. For $k = 0$, this implies that M accepts x iff there is a path in G_0 from $(0, START)$ to $(0, ACCEPT)$. Thus it suffices to show that some $AC^0(\mathsf{STCONN}, \alpha)$ circuit computes the adjacency matrix of each

graph G_k given the input x. In fact, it is easy to see that some AC^0 circuit with input x outputs the adjacency matrix for each edge set E_k^0. Hence it suffices to show that some $AC^0(\mathsf{STCONN}, \alpha)$ circuit computes the adjacency matrix for E_k^1 given input x and the adjacency matrix for E_{k+1}^1, $0 \leq k < h$. This can be done since the elements for the sequence (2.12) can be obtained from E_{k+1} using oracle queries to STCONN, and the string Q that is written on the height $k+1$ oracle tape can be extracted from this sequence using an AC^0 circuit, and so $\alpha(Q)$ can be used to determine u' is the successor of v_t.

Note that the depth of nesting of oracle calls to α is h.

To prove (iii), we note that the direction ($\Leftarrow$) is easy: A $\mathrm{cs}\mathbf{NL}(\alpha)$ machine M on input x answers the single query $f(x, \alpha)$ to STCONN by simulating the $\mathbf{NL}$ machine M' that answers the query on input $f(x, \alpha)$, and each time M' requires another input bit, M deterministically computes that bit.

To prove (iii) in the direction ($\Rightarrow$), we note that the edge relation E_0 defined in the proof of (ii) can be computed by an $\mathrm{cs}\mathbf{L}(\alpha)$ machine. Then as noted above, M accepts its input x iff there is a path in G_0 from $(0, START)$ to $(0, ACCEPT)$, which is an instance of STCONN. $\square$

The next result states an unusual relationship between the relativized and nonrelativized versions of some of our complexity classes. This leaves open the possibility that a proof separating two nonrelativized classes might take advantage of an oracle construction which separates the corresponding relativized classes.

COROLLARY 2.13. *The existence of an oracle α separating any two of the classes*

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

implies the separation of the respective nonrelativized classes.

PROOF.   This follows from Proposition 2.3 and Theorem 2.9 parts (i) and (ii). If any two of the nonrelativized classes are equal, then the corresponding complete problems would be $\mathbf{AC}^0$-equivalent, and hence, the relativized classes would be equal. $\square$

COROLLARY 2.14 (Relativized Immerman-Szelepcsényi Theorem). $cs\mathbf{NL}(\alpha)$ *is closed under complementation.*

PROOF.   By Theorem 2.9 (iii) any language in $co\text{-}cs\mathbf{NL}(\alpha)$ is many-one $cs\mathbf{L}(\alpha)$ reducible to $\overline{\mathsf{STCONN}}$, which is many-one $\mathbf{AC}^0$ reducible to STCONN. So $co\text{-}cs\mathbf{NL}(\alpha) \subseteq cs\mathbf{NL}(\alpha)$.   □

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

PROOF.   The corollary follows from Theorem 2.9 (iii) 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.   □

It is easy to see that the function class associated with either of the classes $\mathbf{AC}^0(\mathbf{1}\text{-}\mathsf{STCONN}, \alpha)$ or $\mathbf{AC}^0(\mathsf{STCONN}, \alpha)$ is closed under composition, so we have the following result.

COROLLARY 2.16. *The function class associated with $cs\mathbf{L}(\alpha)$ is closed under composition.*

However, it is an open question whether the function class associated with $cs\mathbf{NL}(\alpha)$ is necessarily closed under composition, for the same reason that we cannot necessarily conclude that inclusion in part (ii) of Theorem 2.9 can be changed to equality. Once the oracle stack in a $cs\mathbf{NL}(\alpha)$ machine becomes nonempty, it becomes deterministic, so it is not clear that the machine can solve an STCONN problem.

3. Separating the $\mathbf{AC}^k$ hierarchy

One of the obvious benefits of considering relativized complexity classes is that separations are at hand. Even though the unrelativized inclusion of $\mathbf{AC}^1$ in the polynomial hierarchy is strongly conjectured to be strict, no proof is currently known. On the other hand, Wilson (1987) showed the existence of an oracle α_W which

makes the relativized $\mathbf{AC}^k$ hierarchy is strict. Here we reconstruct a technique used by Takeuti (1995) to separate theories in weak bounded arithmetic and use it to give a simpler definition of an oracle α separating the $\mathbf{AC}^k$ hierarchy. In the next section, we show how to use this result together with a witnessing theorem to obtain an unconditional separation of relativized theories capturing the $\mathbf{AC}^k$ hierarchy.

The idea is that computing the k'th iterate $f^k(0) = f(f(\dots f(0)))$ 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 the fact 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 $f \colon \{0,1\}^n \to \{0,1\}^n$.

Of course with such a big domain, we cannot represent such functions simply by a value table. That is 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.[1]

Our argument can be summarized as follows. We assume a circuit of depth d (i.e., the circuit has d levels) is 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 $\ell > d$. 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 $f^i(0)$. 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

---

[1] We use $\log n$ to stand for $\lceil \log_2(n+1) \rceil$.

$\mathrm{dom}(f)$ is a subset of A and its range $\mathrm{rng}(f)$ is a subset of B.

DEFINITION 3.1. A partial function $f \colon \{0,1\}^n \rightharpoonup \{0,1\}^n$ 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 3.1, it is necessarily the case that $0, f(0), f^2(0), \ldots, f^k(0)$ are distinct.

LEMMA 3.2. *Let* $n \in \mathbb{N}$ *and* $f \colon \{0,1\}^n \rightharpoonup \{0,1\}^n$ *be an* ℓ-*sequential partial function. Let* $M \subset \{0,1\}^n$ *be such that* $|\mathrm{dom}(f) \cup M| < 2^n$. *Then there is an* $(\ell + 1)$-*sequential extension* $f' \supseteq f$ *with* $\mathrm{dom}(f') = \mathrm{dom}(f) \cup M$.

PROOF. Let $a \in \{0,1\}^n \backslash (M \cup \mathrm{dom}(f))$. 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 $x \in M \backslash \mathrm{dom}(f)$. 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. $\qquad\square$

DEFINITION 3.3. To any natural number n and any partial function $f \colon \{0,1\}^n \rightharpoonup \{0,1\}^n$, we associate its *bit graph* $\beta_{n,f}$ as a partial function $\beta_{n,f} \colon \{0,1\}^{n+\log n} \rightharpoonup \{0,1\}$ in the obvious way. More precisely, $\beta_{n,f}(xv)$ is the i'th bit of $f(x)$ if $f(x)$ is defined, and undefined otherwise, where v is a string of length $\log n$ coding the natural number i. If $f \colon \{0,1\}^n \to \{0,1\}^n$ is a total function, we define the oracle α_f by

$$\alpha_f(w) \leftrightarrow \beta_{n,f}(w) = 1.$$

Thus $\alpha_f(w)$ can only hold for strings w of length $n + \log n$.

Immediately, from Definition 3.3, we note that f can be uniquely reconstructed from α_f.

If α is an oracle, we define $\alpha^{[n]}$ by

$$\alpha^{[n]}(w) \leftrightarrow (\alpha(w) \wedge |w| = n + \log n),$$

so $\alpha^{[n]}$ has finite support.

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 3.4. *Let C be any circuit of depth d and size strictly less then 2^n. If $C(\alpha)$ correctly computes the last bit of $f^\ell(0)$ for the (uniquely determined) $f\colon \{0,1\}^n \to \{0,1\}^n$ such that $\alpha_f = \alpha^{[n]}$, and this is true for all oracles α, then $\ell \leq d$.*

PROOF.   Assume that such a circuit computes $f^\ell(0)$ correctly for all oracles. We have to find an oracle that witnesses $\ell \leq d$. 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 $k \geq 0$, we define partial functions $f_k\colon \{0,1\}^n \rightharpoonup \{0,1\}^n$ with the following properties. (Here we number the *levels* of the circuit $0, 1, \ldots, d-1$.)

- $f_0 \subseteq f_1 \subseteq f_2 \subseteq \ldots$

- 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.

- β_{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, so f_0 is 0-sequential. 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 3.2.

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

Inspecting the proof of Theorem 3.4, we note that it does not at all use what precisely the nonoracle 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 the above argument and part (ii) of Theorem 2.9.

COROLLARY 3.5. *$csNL(\alpha)$ can iterate a function given by an oracle only constantly far. In particular, there exists α such that $csNL(\alpha)$ 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 3.6. If $g\colon \mathbb{N} \to \mathbb{N}$ is a function from the natural numbers to the natural numbers, and α is an oracle, we define the language

$$\mathcal{L}_g^\alpha = \{x \mid \text{the last bit of } f^{g(n)}(0) \text{ is } 1,$$
$$\text{where } n = |x| \text{ and } f \text{ is such that } \alpha_f = \alpha^{[n]}\}.$$

We note that in Definition 3.6 the function f is uniquely determined by the length n of x and the restriction of α to strings of length $n + \log n$.

Also, for logspace-constructible g the language $\mathcal{L}_g^\alpha$ can be computed by logspace-uniform circuits (with oracle gates) 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 and may have oracle gates.

The language of a circuit family $\{C_n\}_{n\in\mathbb{N}}$ with oracle α is the set of all strings $x \in \{0,1\}^*$ such that the output of $C_{|x|}(\alpha)$ with input x is 1.

DEFINITION 3.7. A *notion of uniformity* is any countable set $\mathcal{U}$ of circuit families.

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

We use a diagonal argument to obtain the following theorem.

THEOREM 3.8. *Let $\mathcal{U}$ be a notion of uniformity and let $\{d_k\}_{k\in\mathbb{N}}$ be a family of functions such that for all $k \in \mathbb{N}$ the function d_{k+1} eventually strictly dominates d_k. Moreover, let $\{s_k\}_{k\in\mathbb{N}}$ be a family of strictly subexponentially growing functions. Then there is a single oracle α that simultaneously witnesses that for all k, $\mathcal{L}^\alpha_{d_{k+1}}$ cannot be computed by any $\mathcal{U}$-uniform $(\mathcal{O}(d_k), \mathcal{O}(s_k))$-circuit family.*

PROOF.   Let $\mathcal{C}^0, \mathcal{C}^1, \ldots$ be an enumeration of $\mathcal{U}$. Let (k_i, c_i, m_i) be an enumeration of all triples of natural numbers.

We will construct natural numbers n_i, and oracles α_i such that the following properties hold.

○ The n_i strictly increase.

○ $\alpha_i(w)$ holds at most for strings w of length $n_i + \log n_i$.

○ If $\mathcal{C}^{m_i} = \{C^{m_i}_n\}_{n\in\mathbb{N}}$ and $C^{m_i}_{n_i}$ has oracle depth at most $c_i \cdot d_{k_i}(n_i)$ and size at most $c_i \cdot s_{k_i}(n_i)$, then the language of $C^{m_i}_{n_i}$ with oracle $\bigvee_{j\leq i} \alpha_j$ differs from $\mathcal{L}^{\bigvee_{j\leq i}\alpha_j}_{d_{k_i+1}}$ at some string of length n_i, and $C^{m_i}_{n_i}$ contains no oracle gates with $n_{i+1} + \log n_{i+1}$ or more inputs.

Then $\bigvee_i \alpha_i$ will be the desired oracle α. For suppose otherwise. Then there is k and a $(\mathcal{O}(d_k), \mathcal{O}(s_k))$ circuit family $\mathcal{C}^m$ such that $\mathcal{C}^m$ with oracle α computes $\mathcal{L}^\alpha_{d_{k+1}}$. Hence there is a triple (k_i, c_i, m_i) such that the circuit $C^{m_i}_{n_i}$ has oracle depth at most $c_i \cdot d_{k_i}(n_i)$ and

size at most $c_i \cdot s_{k_i}(n_i)$ and $C_{n_i}^{m_i}(\alpha)$ correctly computes $\mathcal{L}_{d_{k_i}+1}^{\alpha}$ on inputs of length n_i. But this contradicts the above properties.

At stage i take n_i big enough so that it is bigger than all the previous n_j's and $n_i + \log n_i$ is bigger than the maximal fanin of all the oracle gates in all the circuits looked at so far; moreover, take n_i big enough such that $c_i \cdot s_{k_i}(n_i) < 2^{n_i}$ and $c_i \cdot d_{k_i}(n_i) < d_{k_i+1}(n_i)$. This is possible as s_{k_i} has strictly subexponential growth and d_{k_i+1} dominates d_{k_i} eventually.

Look at the n_i'th circuit in the circuit family $\mathcal{C}^{m_i}$, and call it C. We may assume that C has oracle depth at most $c_i \cdot d_{k_i}(n_i)$ and size at most $c_i \cdot s_{k_i}(n_i)$ for otherwise there is nothing to show and we can choose α_i to be the empty set.

By Theorem 3.4, we can find an oracle α_i whose support includes only strings of length $n_i + \log n_i$ such that C with oracle $\bigvee_{j \leq i} \alpha_i$ does not solve the decision problem associated with $f^{d_{k_i+1}(n_i)}(0)$ for f given by $\alpha_f = \alpha_i$. $\qquad\square$

COROLLARY 3.9. *There is a single oracle $\alpha \subseteq \{0,1\}^*$ for which*

$$(3.10) \qquad \mathbf{AC}^k(\alpha) \subseteq \mathbf{NC}^{k+1}(\alpha) \subsetneq \mathbf{AC}^{k+1}(\alpha), \text{ for all } k \geq 1$$

and $cs\mathbf{NL}(\alpha) \subsetneq \mathbf{AC}^1(\alpha)$.

PROOF.   In Theorem 3.8, let $\mathcal{U}$ be logspace uniformity and let $d_k(n) = \log^k n$ and $s_k(n) = 2^{\lceil n/2 \rceil}$. Then for $k \geq 1$, every problem in $\mathbf{AC}^k(\alpha)$ can be computed by a $\mathcal{U}$-uniform $(\mathcal{O}(d_k), \mathcal{O}(s_k))$-circuit family with oracle α, and $\mathcal{L}_{d_k}^{\alpha}$ is in $\mathbf{AC}^k(\alpha)$. Then Theorem 3.8 shows that there is a single oracle α satisfying (3.10).

To show $cs\mathbf{NL}(\alpha) \subsetneq \mathbf{AC}^1(\alpha)$, by Theorem 2.9 part (ii), it suffices to show

$$(3.11) \qquad\qquad \mathcal{L}_{d_1}^{\alpha} \notin \mathbf{AC}^0(\mathsf{STCONN}, \alpha).$$

Theorem 3.8 shows how to construct α so $\mathcal{L}_{d_1}^{\alpha} \notin \mathbf{AC}^0(\alpha)$, and we can modify the proof by starting with α which is a version of STCONN in which no string has length $n + \log n$ for any n. The result of the construction in the proof satisfies (3.11) as well as (3.10). $\qquad\square$

4. Theories for relativized classes

4.1. Two-sorted languages and complexity classes. Our
theories are based on a two-sorted vocabulary, and it is convenient
to re-interpret the complexity classes using this vocabulary (Cook
& Nguyen 2010). Our two-sorted language has variables $x, y, z, \ldots$
ranging over $\mathbb{N}$ and variables $X, Y, Z, \ldots$ 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 corre-
sponding 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 Krajíček 1995) 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 vocabu-
lary $\mathcal{L}$ 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 func-
tions $F(\vec{x}, \vec{X})$), 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 (Immerman 1999), including $\mathbf{LTH}$ (the log time hierarchy on alternating Turing machines) and $\mathbf{FO}$ (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 $\mathbf{FO}$ characterization of $\mathbf{AC}^0$ we obtain the following nice connection between the classes $\mathbf{AC}^0$ and $\mathbf{NP}$ and our two-sorted $\mathcal{L}_A^2$-formulas (see Theorems IV.3.6 and IV.3.7 in Cook & Nguyen 2010).

THEOREM 4.1 (Representation Theorem). *A relation $R(\vec{x}, \vec{X})$ is in* $\mathbf{AC}^0$ *(resp. $\mathbf{NP}$) iff it is represented by some Σ_0^B (resp. Σ_1^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* $B_F(i, \vec{x}, \vec{X})$ to hold iff the ith bit of $F(\vec{x}, \vec{X})$ is one. Formally

$$(4.2) \qquad B_F(i, \vec{x}, \vec{X}) \leftrightarrow F(\vec{x}, \vec{X})(i).$$

(Compare this with Definition 3.3.)

DEFINITION 4.3. *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 4.4. *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 $\mathbf{FAC}^0$. Of course the set of functions in $\mathbf{FAC}^0$ is closed under composition, but for a general vocabulary $\mathcal{L}$ the set of functions $\mathbf{\Sigma}_0^B$-definable from $\mathcal{L}$ may not be closed under composition. For example, if a relation $\alpha(Y)$ codes the bit graph of a function F, then F could be $\mathbf{\Sigma}_0^B$-definable from $\mathcal{L} \cup \{\alpha\}$ but $F \circ F$ may not be. 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 4.5. *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*
(4.6)
$$F_i \text{ is } \mathbf{\Sigma}_0^B\text{-definable from } \mathcal{L} \cup \{F_1, \ldots, F_{i-1}\}, \text{ for } i = 1, \ldots, n;$$

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.*

If F and G are string-to-string functions in $\mathcal{L}$, then the term $F(G(X))$ can appear in $\mathbf{\Sigma}_0^B(\mathcal{L})$ formulas, so the set of functions $\mathbf{AC}^0$-reducible to $\mathcal{L}$ is always closed under composition. In fact, from the techniques used to prove Theorem 4.1, we can show that F is $\mathbf{AC}^0$-reducible to $\mathcal{L}$ iff 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).

The (two-sorted) classes $\mathbf{AC}^0(m), \mathbf{TC}^0, \mathbf{NC}^1, \mathbf{L}$ and $\mathbf{NL}$ are the closuresunder $\mathbf{AC}^0$reductions of their respective complete

problems, so they become $\mathbf{AC}^0(\mathsf{MOD}m)$, $\mathbf{AC}^0(\mathsf{THRESH})$, $\mathbf{AC}^0$ ($\mathsf{FORMVAL}$), $\mathbf{AC}^0(\mathbf{1}\text{-}\mathsf{STCONN})$ and $\mathbf{AC}^0(\mathsf{STCONN})$. The relativized versions $\mathbf{AC}^0(\alpha)$, $\mathbf{AC}^0(m,\alpha)$, $\mathbf{TC}^0(\alpha)$, $\mathbf{NC}^k(\alpha)$, and $\mathbf{AC}^k(\alpha)$ of the circuit classes are all closed under $\mathbf{AC}^0$ reductions, and so is $\mathrm{cs}\mathbf{L}(\alpha)$ (Theorem 2.9 part (i)), but $\mathrm{cs}\mathbf{NL}(\alpha)$ may not be so closed.

4.2. Nonrelativized theories.   In this paper, we consider theories $\mathcal{T}$ over two-sorted vocabularies which contain $\mathcal{L}_A^2$.

DEFINITION 4.7. *If $F(\vec{x}, \vec{X})$ is a string function, we say that F is $\mathbf{\Sigma}_1^B$-definable (or provably total) in $\mathcal{T}$ if there is a $\mathbf{\Sigma}_1^B$ formula $\varphi(\vec{x}, \vec{X}, Y)$ which represents the graph of F and*

$$\mathcal{T} \vdash \exists! Y \varphi(\vec{x}, \vec{X}, Y)$$

where $\exists! Y$ means there exists a unique Y.

A similar definition applies to number functions $f(\vec{x}, \vec{X})$. When we associate a theory $\mathcal{T}$ with a complexity class $\mathbf{C}$, we want the provably total functions in $\mathcal{T}$ to coincide with the functions in $\mathbf{FC}$.

The theory $\mathbf{V}^0$ (essentially $\mathbf{\Sigma}_0^p\text{-}comp$ in Zambella 1996, and $\mathbf{I\Sigma}_0^{1,b}$ (without $\#$) in Krajíček 1995) is the theory over $\mathcal{L}_A^2$ that is axiomatized by the axioms listed in Figure 4.1 together with the comprehension axiom scheme $\mathbf{\Sigma}_0^B\text{-}\mathbf{COMP}$, i.e., the set of all formulas of the form

$$(4.8) \qquad \exists X \leq y \forall z < y (X(z) \leftrightarrow \varphi(z)),$$

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. $X(y) \supset y < |X|$	**L2.** $y + 1 = |X| \supset X(y)$
SE. $[|X| = |Y| \wedge \forall i < |X|(X(i) \leftrightarrow Y(i))] \supset X = Y$	

Figure 4.1: 2-**BASIC**

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

We associate $\mathbf{V}^0$ with the complexity class $\mathbf{AC}^0$, and indeed, the provably total functions of $\mathbf{V}^0$ comprise the class $\mathbf{FAC}^0$. All theories considered in this paper extend $\mathbf{V}^0$.

In Cook & Nguyen (2010), Chapter IX, for various subclasses $\mathbf{C}$ of $\mathbf{P}$, a theory $\mathbf{VC}$ is developed which is associated with $\mathbf{C}$ as above, so the provably total functions of $\mathbf{VC}$ are precisely those in $\mathbf{FC}$. Essentially, the theory $\mathbf{VC}$ is axiomatized by the axioms of $\mathbf{V}^0$ together with an axiom that states the existence of a polytime computation for a complete problem of $\mathbf{C}$, assuming the parameters as given inputs. The additional axioms for the classes of interest in this paper will be listed below. For the logspace classes (i.e., $\mathbf{AC}^0(m), \ldots, \mathbf{NL}$), we use roughly the same problems as those mentioned in the previous sections. For other classes in the $\mathbf{AC}$ hierarchy, we use the monotone circuit value problem, with appropriate restrictions on the depth and fanin of the circuits.

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

$$(4.9) \qquad\qquad X(y, z) \equiv X(\langle y, z \rangle).$$

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 $0 \le u, v < a$.

We will also use the number function $(Z)^x$ which is the x-th element of the sequence of numbers encoded by Z:

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

In addition, $\log a$, or $|a|$, denotes the integral part of $\log_2(a + 1)$. Note that these functions are provably total in $\mathbf{V}^0$. Note also that the functions $(Z)^x$ and $|a|$ can be eliminated using their $\mathbf{\Sigma}_0^B$ defining axioms, so that $\mathbf{\Sigma}_0^B$ formulas that contain these functions are in fact equivalent to $\mathbf{\Sigma}_0^B$ formulas over $\mathcal{L}_A^2$ (see Cook & Nguyen 2010, Lemma V.4.15.)

We now list the additional axioms for the classes considered in this paper. (Recall that the base theory is always $\mathbf{V}^0$.) First, consider $\mathbf{TC}^0$. The theory $\mathbf{VTC}^0$ is axiomatized by the axioms of $\mathbf{V}^0$ and the following axiom:

$$NUMONES \equiv \exists Y \le 1 + \langle x, x \rangle \delta_{NUM}(x, X, Y)$$

where

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

Here $NUMONES$ formalizes a computation of the number of 1-bits in the string $X(0), X(1), \dots, X(x-1)$: for $1 \le z \le x$, $(Y)^z$ is the number of 1 bits in $X(0), X(1), \dots, X(z-1)$. (Note that computing the number of 1-bits in the input string is roughly the same as computing the threshold function.)

Now consider $\mathbf{AC}^0(m)$. The additional axiom for the theory $\mathbf{V}^0(m)$ associated with $\mathbf{AC}^0(m)$ is

$$MOD_m \equiv \exists Y \delta_{MOD_m}(x, X, Y)$$

where

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

Similar to $NUMONES$ above, here $(Y)^z$ is the number of 1-bits in the sequence $X(0), \dots, X(z-1)$ modulo m.

For $\mathbf{NC}^1$, the additional axiom MFV for $\mathbf{VNC}^1$ states the existence of a polytime computation for the balanced monotone sentence value problem which is complete for $\mathbf{VNC}^1$ (Buss 1987):

$$(4.10) \qquad MFV \equiv \exists Y \, \delta_{MFVP}(a, G, I, Y)$$

where

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

Here the balanced monotone sentence is viewed as a balanced binary tree encoded by (a, G) and I specifies the leaves of the tree: node x's children are $2x$ and $2x + 1$, $G(x)$ indicates whether node x is an $\vee$ or $\wedge$ node, and $I(z)$ is the value of leaf z. Y is the bottom-up evaluation of the sentence: $Y(x)$ is the value of node x.

Next, the theory **VNL** for **NL** is axiomatized by $\mathbf{V}^0$ and the following axiom

$$CONN \equiv \exists Y \, \delta_{CONN}(a, E, Y),$$

where $\delta_{CONN}(a, E, Y)$ states that Y encodes a polytime computation for the following problem, which is equivalent to STCONN under $\mathbf{AC}^0$-many-one reductions: on input (a, E) which encodes a directed graph, compute the nodes that are reachable from node 0. Here $Y(z, x)$ holds iff there is a path from 0 to x of length $\leq z$; more precisely,

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

For **L**, the additional axiom in **VL** is $PATH$, which states the existence of a polytime computation for the following problem, which is equivalent to **1**-STCONN under $\mathbf{AC}^0$ reductions: on input a directed graph with out degree at most one which is encoded by (a, E), compute the transitive closure of vertex 0:

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

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

Now we consider the classes $\mathbf{AC}^k$ for $k \geq 1$. We use the monotone circuit value problem under an appropriate setting. In particular, the circuit has unbounded fanin, and its depth is $(\log n)^k$, where n is the number of its inputs. Thus the additional axiom in $\mathbf{VAC}^k$ states the existence of a polytime evaluation Y for a circuit of this kind which is encoded by (a, E, G, I):

$$(4.11) \qquad\qquad \exists Y \, \delta_{LMCV}(a, |a|^k, E, G, I, Y),$$

where the depth parameter d is set to $|a|^k$ in the formula

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

The formula $\delta_{LMCV}(w, d, E, G, I, Y)$ (layered monotone circuit value) states that Y is an evaluation of the circuit encoded by (w, d, E, G) on input I. The circuit is encoded as follows. There are $(d + 1)$ layers in the circuit, each of them contains w gates. Hence each gate is given by a pair (z, x) where z indicates the layer (inputs to the circuits are on layer 0 and outputs are on layer d), and x is the position of the gate on that layer. E specifies the wires in the circuit: $E(z, u, x)$ holds if and only if gate (z, u) is an input to gate $(z + 1, x)$, and G specifies the gates: $G(z, x)$ holds if gate (z, x) is an $\wedge$ gate, otherwise it is an $\vee$ gate. Bit $Y(z, x)$ is the value of gate (z, x).

For $\mathbf{NC}^k$ ($k \geq 2$), the circuit value problem is restricted further, so that the circuit's fanin is at most 2. We express this condition by the formula $Fanin2(w, d, E)$, where (w, d, E) encodes the underlying graph of the circuit as above:

$$Fanin2(w, d, E) \equiv \forall z < d \forall x < w \exists u_1 < w \exists u_2 < w \forall v < w$$
$$E(z, v, x) \supset (v = u_1 \vee v = u_2).$$

Similar to $\mathbf{VAC}^k$, the theory $\mathbf{VNC}^k$ (for $k \geq 2$) is axiomatized by the axioms of $\mathbf{V}^0$ together with

(4.12) 　　$(Fanin2(a, |a|^k, E) \supset \exists Y \, \delta_{LMCV}(a, |a|^k, E, G, I, Y)).$

The connection between the above theories $\mathbf{VAC}^k$ and $\mathbf{VNC}^k$ and their corresponding classes is discussed in detail in Cook & Nguyen (2010), Section IX.5.6. The key point for these theories (as well as the others) is that the problem of witnessing the existential quantifiers in the axiom for each theory (in this case (4.11) and (4.12)) is complete for the associated complexity class. For $\mathbf{AC}^k$ and $\mathbf{NC}^k$ this is shown by using the characterization of these classes in terms of alternating Turing machines.

In the remainder of this section, we will present relativized theories $\mathbf{VC}(\alpha)$ that characterize the relativized classes discussed in Section 2.

4.3. Relativized theories.

4.3.1. Classes with bounded nested oracle depth.
We first look at $\mathbf{AC}^0(m,\alpha)$, $\mathbf{TC}^0(\alpha)$, $\mathbf{NC}^1(\alpha)$ and $\mathrm{cs}\mathbf{L}(\alpha)$. These classes have constant nested depth of oracle gates, and they are the $\mathbf{AC}^0$-closure of the oracle α and an appropriate complete problem (see Proposition 2.3 and Theorem 2.9 (i)). We can treat these as the $\mathbf{AC}^0(\alpha)$-closure of the complete problem for their respective non-relativized version. Thus the development in Cook & Nguyen (2010), Chapters VIII and IX, can be readily extended to these classes. The change we need to make here is to replace the base theory $\mathbf{V}^0$ by its relativized version, $\mathbf{V}^0(\alpha)$, which is axiomatized by comprehension axioms (4.8) over $\mathbf{\Sigma}_0^B(\alpha)$ formulas instead of just $\mathbf{\Sigma}_0^B$ formulas.

First note that a sequence of strings can be encoded using the string function Row, where $Row(x, Z)$ extracts row x from the array coded by Z. Thus

$$(4.13) \qquad Row(x, Z)(i) \leftrightarrow i < |Z| \wedge Z(x, i).$$

We will also write $Z^{[x]}$ for $Row(x, Z)$.

Notation. For a predicate α we use $\mathcal{L}_A^2(\alpha)$ to denote $\mathcal{L}_A^2 \cup \{Row, \alpha\}$, and we use $\mathbf{\Sigma}_0^B(\alpha)$ and $\mathbf{\Sigma}_1^B(\alpha)$ to denote the classes $\mathbf{\Sigma}_0^B(Row, \alpha)$ and $\mathbf{\Sigma}_1^B(Row, \alpha)$, respectively. Definitions 4.4, 4.5, and 4.7 of $\mathbf{\Sigma}_0^B$-definable, $\mathbf{AC}^0$-reducible, and $\mathbf{\Sigma}_1^B$-definable in a theory, are extended in the obvious way to $\mathbf{\Sigma}_0^B(\alpha)$-definable, $\mathbf{AC}^0(\alpha)$-reducible, and $\mathbf{\Sigma}_1^B(\alpha)$-definable in a theory.

Atomic formulas containing α have the form $\alpha(T)$, where T is a string term, namely either a variable X or a term $Row(t, T')$ for terms t, T'.

Notice that while the string function Row can occur nested in a $\mathbf{\Sigma}_0^B(\alpha)$ formula, the predicate α cannot. Thus a $\mathbf{\Sigma}_0^B(\alpha)$ formula represents relations computable by a family of polynomial size $\mathbf{AC}^0(\alpha)$ circuits whose oracle nested depth is one.

The function *Row* is useful in constructing formulas describing circuits which query the oracle α. For example, if an n-ary gate g has inputs from n different α gates, we can code the sequence of inputs to the α gates using a string X, so the ith input bit to g is $\alpha(X^{[i]})$.

DEFINITION 4.14. *The following theories have vocabulary $\mathcal{L}_A^2(\alpha)$ and include the defining axiom (4.13) for Row.* $\mathbf{V}^0(\alpha) = \mathbf{V}^0 + \mathbf{\Sigma}_0^B(\alpha)\text{-}\mathbf{COMP}$. *The theories* $\mathbf{V}^0(m, \alpha), \mathbf{VTC}^0(\alpha), \mathbf{VNC}^1(\alpha)$ *and* $\mathbf{VL}(\alpha), \mathbf{VNL}'(\alpha)$ *are axiomatized by the axioms of* $\mathbf{V}^0(\alpha)$ *together with the axiom:* MOD_m, $NUMONES$, MFV, $PATH$, $CONN$, *respectively. (See Section 4.2.)*

Note that equality axioms (implicitly) hold for the new symbols *Row* and α.

The next result connects the theories with their corresponding complexity classes, except for the theory $\mathbf{VNL}'(\alpha)$, which corresponds to the class $\mathbf{AC}^0(\mathsf{STCONN}, \alpha)$ (see part (ii) of Theorem 2.9) rather than cs$\mathbf{NL}(\alpha)$. We are not able to provide a theory exactly associated with cs$\mathbf{NL}(\alpha)$ because we cannot show that the associated function class is closed under composition.

The first step in the proof of the next theorem is to show that a function is in $\mathbf{FAC}^0(\alpha)$ iff it is $\mathbf{\Sigma}_1^B(\alpha)$-definable in $\mathbf{V}^0(\alpha)$. For the direction ($\Rightarrow$), the proof of the unrelativized case uses the $\mathbf{\Sigma}_0^B$ Representation Theorem (Theorem 4.1), but that result does not hold for the relativized case, because, as remarked above, $\mathbf{\Sigma}_0^B(\alpha)$ formulas only represent $\mathbf{AC}^0(\alpha)$ relations that can be computed by circuits of oracle depth one.

So results in Chapter IX of Cook & Nguyen (2010) are required.

THEOREM 4.15. *For a class* $\mathbf{C}$ *in* $\{\mathbf{AC}^0, \mathbf{AC}^0(m), \mathbf{TC}^0, \mathbf{NC}^1, \mathbf{L}\}$, *a function is in* $\mathbf{FC}(\alpha)$ *if and only if it is* $\mathbf{\Sigma}_1^B(\alpha)$-*definable in* $\mathbf{VC}(\alpha)$.

PROOF.   We start by proving this when $\mathbf{C}$ is $\mathbf{AC}^0$: A function is in $\mathbf{FAC}^0(\alpha)$ iff it is $\mathbf{\Sigma}_1^B(\alpha)$-definable in $\mathbf{V}^0(\alpha)$. The 'if' direction follows from a standard witnessing theorem (see, for example, Chapter V in Cook & Nguyen 2010), because the existential quantifier in each $\mathbf{\Sigma}_0^B(\alpha)\text{-}\mathbf{COMP}$ axiom is witnessed by an $\mathbf{FAC}^0(\alpha)$ function whose graph is represented by a $\mathbf{\Sigma}_0^B(\alpha)$ formula.

The converse follows from a slight generalization of Theorem IX.2.3 in Cook & Nguyen (2010), where the original states that a function is Σ_1^B-definable in **VC** iff it is in **FC**. That theorem applies to complexity classes **C** consisting of the relations $\mathbf{AC}^0$-reducible to a string function $F(X)$ whose graph is represented by a Σ_0^B-formula $\delta_F(X, Y)$. The theory **VC** has vocabulary $\mathcal{L}_A^2$ and is axiomatized by the axioms of $\mathbf{V}^0$ together with

$$(4.16) \qquad\qquad \exists Y \le b \forall i < b \delta_F(X^{[i]}, Y^{[i]}).$$

The generalization we need (which is proved in the same way) is that the theory **VC** is replaced by a theory $\mathbf{VC}(\alpha)$ with vocabulary $\mathcal{L}_A^2(\alpha)$ axiomatized by the axioms of $\mathbf{V}^0(\alpha)$ and (4.16), where now $\delta_F(X, Y)$ is a $\Sigma_0^B(\alpha)$-formula. The assertion now is that a function is $\Sigma_1^B(\alpha)$-definable in $\mathbf{VC}(\alpha)$ iff it is in $\mathbf{FC}(\alpha)$.

To apply this to the theory $\mathbf{V}^0(\alpha)$ we take $F = F_\alpha$ where $\delta_{F_\alpha}(X, Y)$ is the formula $|Y| \le |X| \wedge \forall j < |X|(Y(j) \leftrightarrow \alpha(X^{[j]}))$. Thus $F_\alpha(X)$ is the bit string resulting from applying α successively to the elements of the sequence of strings coded by X, and so the $\mathbf{AC}^0$ closure of F_α is $\mathbf{FAC}^0(\alpha)$. The theory $\mathbf{V}^0(\alpha)$ has the comprehension axiom $\Sigma_0^B(\alpha)$-**COMP**, which implies (4.16) when F is taken to be F_α. Thus our generalized Theorem IX.2.3 in Cook & Nguyen (2010) implies that every function in $\mathbf{FAC}^0(\alpha)$ is $\Sigma_1^B(\alpha)$-definable in $\mathbf{V}^0(\alpha)$.

Theorem 4.15 for the other complexity classes follows from Proposition 2.3 and Theorem 2.9 (i) and the fact that the theories for the nonrelativized classes capture the nonrelativized classes (Section 4.2). $\square$

The same argument shows that the theory $\mathbf{V}^0(\alpha) + CONN$ is associated with the class $\mathbf{AC}^0(\mathsf{STCONN}, \alpha)$. But this class might not be the same as $\mathrm{cs}\mathbf{NL}(\alpha)$. In fact, as pointed out after the proof of Corollary 2.16, the function class associated with $\mathrm{cs}\mathbf{NL}(\alpha)$ may not be closed under composition, and hence, $\mathrm{cs}\mathbf{NL}(\alpha)$ may not be closed under $\mathbf{AC}^0$ reductions, so the framework of Cook & Nguyen (2010), Chapter IX, may not apply to this class.

Notice also that we can relativize the axioms MOD_m, $NUMONES$, MFV and $PATH$ in the obvious way, i.e., by replacing the string variables X in $NUMONES$ and MOD_m, G and I in

MFV, and E in $PATH$ by $\mathbf{\Sigma}_0^B(\alpha)$ formula(s). It turns out that these relativized axioms are provable in the respective relativized theories, and in fact, they can be used together with $\mathbf{V}^0$ to axiomatize the theories. More specifically, let $NUMONES(\alpha)$ denote the following *axiom scheme*:

(4.17) $\exists Y \leq 1 + \langle x, x \rangle \; (Y)^0 = 0 \; \wedge$
$\forall z < x \; \left[(\varphi(z) \supset (Y)^{z+1} = (Y)^z + 1) \wedge (\neg\varphi(z) \supset (Y)^{z+1} = (Y)^z) \right]$

for all $\mathbf{\Sigma}_0^B(\alpha)$ formulas φ that do not contain Y. Similarly, we can define $MOD_m(\alpha)$, $MFV(\alpha)$, and $PATH(\alpha)$.

The next result is useful in the next subsection.

PROPOSITION 4.18. $\mathbf{VTC}^0(\alpha)$ *can be equivalently axiomatized by the axioms of* $\mathbf{V}^0$ *and* $NUMONES(\alpha)$. *Similarly for* $\mathbf{V}^0(m, \alpha)$, $\mathbf{VNC}^1(\alpha)$, *and* $\mathbf{VL}(\alpha)$, *with* $NUMONES(\alpha)$ *replaced, respectively, by* $MOD_m(\alpha)$, $MFV(\alpha)$ *and* $PATH(\alpha)$.

PROOF.   We prove this for $NUMONES(\alpha)$. The other cases are similar.

It is relatively simple to show that the axioms of $NUMONES(\alpha)$ are provable in $\mathbf{VTC}^0(\alpha)$. Indeed, consider an axiom in $NUMONES(\alpha)$ as in (4.17) above. By $\mathbf{\Sigma}_0^B(\alpha)$-**COMP**, there is a string X such that $X(z) \leftrightarrow \varphi(z)$ for all $z < x$. Hence, the string Y that satisfies $\delta_{NUM}(x, X, Y)$ satisfies (4.17).

For the other direction, suppose that we want to prove the following instance of $\mathbf{\Sigma}_0^B(\alpha)$-**COMP** using $NUMONES(\alpha)$ and $\mathbf{V}^0$:

$$\exists Z \leq b \forall z < b, \; Z(z) \leftrightarrow \varphi(z),$$

where φ is a $\mathbf{\Sigma}_0^B(\alpha)$ formula. Using $NUMONES(\alpha)$ we obtain a string Y as in (4.17) (for $x = b$). Now, it is straightforward to identify those $z < b$ such that $\varphi(z)$ holds:

$$\varphi(z) \leftrightarrow (Y)^{z+1} = (Y)^z + 1.$$

Thus Z can be defined using $\mathbf{\Sigma}_0^B$-**COMP** from Y.

The arguments for $\mathbf{V}^0(m, \alpha)$, $\mathbf{VNC}^1(\alpha)$ and $\mathbf{VL}(\alpha)$ are similar.
$\square$

4.3.2. Classes with unbounded oracle nested depth.   Now
we present the theories $\mathbf{VAC}^k(\alpha)$ (for $k \geq 1$) and $\mathbf{VNC}^k(\alpha)$ (for
$k \geq 2$). For the nonrelativized case, the axioms for the theories
use the fact that the problem of evaluating an unbounded fanin
(resp. bounded fanin) circuit of depth $(\log n)^k$ is $\mathbf{AC}^0$-complete
for $\mathbf{AC}^k$ (resp. $\mathbf{NC}^k$).[2] Unfortunately, these nonrelativized prob-
lems are not $\mathbf{AC}^0(\alpha)$-complete for the corresponding relativized
problems, unlike the situation for the classes with bounded oracle
nested depth considered previously. However, for the oracle ver-
sions of the circuit classes, the evaluation problems become $\mathbf{AC}^0$-
complete for the corresponding relativized classes, provided (in the
case of $\mathbf{NC}^k$) the circuit descriptions tell the nested oracle depth
of each oracle gate. Thus $\mathbf{VAC}^k(\alpha)$ (or $\mathbf{VNC}^k(\alpha)$) will be axiom-
atized by $\mathbf{V}^0$ together with an additional axiom that formalizes an
oracle computation that solves the respective complete problem.

First we describe the encoding of the input. As before, a circuit
of width w and depth d will be encoded by (w, d, E, G), and its
input will be denoted by I. Since the order of inputs to an oracle
gate is important, the string variable E that encodes the wires
in the circuit is now four-dimensional: $E(z, u, t, x)$ indicates that
gate (z, u) (i.e., the u-th gate on layer z) is the t-th input to gate
$(z + 1, x)$. Also, the type of a gate (z, x) is specified by $(G)^{\langle z,x\rangle}$ as
before, but now it can have value in $\wedge$, $\vee$, $\neg$, or α (we no longer
consider just monotone circuits). We use the following formula to
ensure that this is a valid encoding; it says that each gate $(z + 1, x)$
has an arity s which is 1 if the gate is a $\neg$-gate. Moreover, for $t < s$
the t-th input to a gate is unique.

$$Proper(w, d, E, G) \equiv \forall z < d \ \forall x < w \ \exists! s \leq w$$
$$(s \geq 1 \wedge Arity(z + 1, x, s, E)) \wedge$$
$$(G)^{\langle z+1,x\rangle} = \text{``}\neg\text{''} \supset Arity(z + 1, x, 1, E),$$

where $Arity(z + 1, x, s, E)$ (which asserts that gate $(z + 1, x)$ has
arity s) is the formula:
(4.19)
$$\forall t < s \exists! u < w \ E(z, u, t, x) \wedge \forall t < w(s \leq t \supset \neg \exists u < wE(z, u, t, x)).$$

---

[2]For $\mathbf{NC}^1$ the circuit must be presented in a transparent way, such as a
balanced monotone formula as in the axiom (4.10) for $\mathbf{VNC}^1$.

The formula $\delta^\alpha_{LOCV}(w, d, E, G, I, Q, Y)$ defined below states that (Q, Y) is an evaluation of the oracle circuit (w, d, E, G) on input I. Here the string $Q^{[z+1,x]}$ encodes the query to the oracle gate $(z+1, x)$, and bit $Y(z, x)$ is the value of gate (z, x). (LOCV stand for 'layered oracle circuit value.')

DEFINITION 4.20. *The formula* $\delta^\alpha_{LOCV}(w, d, E, G, I, Q, Y)$ *is the formula*

$$
\begin{aligned}
\forall z < d \, \forall x < w, \quad & [Y(0, x) \leftrightarrow I(x)] \wedge \\
& \big[\forall t < w (Q^{[z+1,x]}(t) \leftrightarrow (\exists u < w, \, E(z, u, t, x) \wedge Y(z, u)))\big] \wedge \\
\big[Y(z+1, x) \leftrightarrow & \big(((G)^{\langle z+1, x\rangle} = \text{``}\wedge\text{''} \wedge \forall t, u < w, \, E(z, u, t, x) \supset Y(z, u)) \vee \\
& ((G)^{\langle z+1, x\rangle} = \text{``}\vee\text{''} \wedge \exists t < w \exists u < w, \, E(z, u, t, x) \wedge Y(z, u)) \vee \\
& ((G)^{\langle z+1, x\rangle} = \text{``}\neg\text{''} \wedge \exists u < w, \, E(z, u, 0, x) \wedge \neg Y(z, u)) \vee \\
& ((G)^{\langle z+1, x\rangle} = \text{``}\alpha\text{''} \wedge \alpha(Q^{[z+1,x]})))\big].
\end{aligned}
$$

DEFINITION 4.21 ($\mathbf{VAC}^k(\alpha)$). *For* $k \geq 1$, $\mathbf{VAC}^k(\alpha)$ *is the theory over the vocabulary* $\mathcal{L}^2_A(\alpha)$ *and is axiomatized by the axioms of* $\mathbf{V}^0$ *and the following axiom:*
(4.22)
$$(Proper(w, d, E, G) \supset \exists Q \exists Y \, \delta^\alpha_{LOCV}(w, |w|^k, E, G, I, Q, Y)).$$

The axiom (4.22) asserts that $\mathbf{VAC}^k$ circuits can be evaluated. Since a function in $\mathbf{FAC}^k(\alpha)$ is computed by an $\mathbf{AC}^0$-uniform family of $\mathbf{AC}^k(\alpha)$ circuits, and our method of describing an oracle circuit by the tuple (w, d, E, G) can be taken as a definition, the axiom is clearly strong enough to show that the functions in $\mathbf{FAC}^k(\alpha)$ are $\Sigma^B_1(\alpha)$-definable in $\mathbf{VAC}^k(\alpha)$. But in order to show the converse, we need to show that the existential quantifiers $\exists Q \exists Y$ can be witnessed by functions in $\mathbf{FAC}^k(\alpha)$, and for this, we need to show the existence of universal circuits for $\mathbf{AC}^k(\alpha)$. This is done in the next result.

PROPOSITION 4.23. *For* $k \geq 1$, *the problem of evaluating the circuit encoded by* $(w, |w|^k, E, G)$ *on a given input* I, *assuming*

Proper$(w, |w|^k, E, G)$ *is satisfied, is complete for* $\mathbf{FAC}^k(\alpha)$ *under* $\mathbf{AC}^0$*-many-one reductions.*

PROOF.   The hardness direction follows by the discussion above: every function $F(X)$ in $\mathbf{FAC}^k(\alpha)$ can be computed by a circuit family in which the parameters w, E, G, I for each circuit can be computed by $\mathbf{AC}^0$ functions of X.

Conversely, we need to prove membership of the circuit evaluation problem in $\mathbf{FAC}^k(\alpha)$. We do this for the case $k = 1$. The proof for the general case is similar. Thus we need to construct a universal circuit for oracle circuits of depth $\log n$. In fact, we will construct a universal circuit (of depth $\mathcal{O}(d)$, size polynomial in w, d) for all circuits of depth d and width w. Let $C = (w, d, E, G)$ denote the given circuit. The idea is to construct a component $K_{z,x}$ for each gate (z, x) in C, where $z < d$ and $x < w$: $K_{z+1,x}$ is an $\mathbf{AC}^0(\alpha)$ circuit that takes inputs from E, G, I and $K_{z,u}$ for all $u < w$, so that when each $K_{z,u}$ computes gate (z, u) in C, $K_{z+1,x}$ computes the value of gate $(z + 1, x)$. We will present $K_{z,x}$ as a bounded depth formula.

The circuits $K_{0,x}$ are easy to define: for all $x < w$, $K_{0,x} \equiv I(x)$. For $z \geq 0$, $K_{z+1,x}$ is the following disjunction:

$$(G)^{\langle z+1,x \rangle} = \text{``}\wedge\text{''} \wedge \bigwedge_{t<w} \bigwedge_{u<w} E(z, u, t, x) \supset K_{z,u} \vee$$

$$(G)^{\langle z+1,x \rangle} = \text{``}\vee\text{''} \wedge \bigvee_{t<w} \bigvee_{u<w} E(z, u, t, x) \wedge K_{z,u} \vee$$

$$(G)^{\langle z+1,x \rangle} = \text{``}\neg\text{''} \wedge \bigvee_{u<w} E(z, u, 0, x) \wedge \neg K_{z,u} \vee$$

$$\left[(G)^{\langle z+1,x \rangle} = \text{``}\alpha\text{''} \wedge \bigvee_{s<w} \left(Arity(z, x, s, E) \right. \right.$$

$$\wedge\, \alpha(\bigvee_{u<w} (E(z, u, 0, x) \wedge K_{z,u}), \ldots,$$

$$\bigvee_{u<w} (E(z, u, s-1, x) \wedge K_{z,u}))) \Big].$$

Now by arranging $K_{z,x}$ in the same order as (z, x), we obtain an $\mathbf{AC}^1(\alpha)$ circuit that evaluates C. $\square$

THEOREM 4.24. *For $k \geq 1$, the functions in $\mathbf{FAC}^k(\alpha)$ are precisely the $\Sigma_1^B(\alpha)$-definable functions of $\mathbf{VAC}^k(\alpha)$.*

PROOF.   By Proposition 4.23, the problem of witnessing the quantifiers $\exists Q \exists Y$ in the axiom (4.22) is in $\mathbf{FAC}^k(\alpha)$, and so by a standard witnessing argument every $\Sigma_1^B(\alpha)$-definable function is in $\mathbf{FAC}^k(\alpha)$. The converse follows from the hardness direction of Proposition 4.23 and the fact that the $\Sigma_1^B(\alpha)$-definable functions in $\mathbf{VAC}^k(\alpha)$ are closed under $\mathbf{AC}^0$ reductions, by the methods used in Chapter IX of Cook & Nguyen (2010).   □

Finally, we consider $\mathbf{NC}^k(\alpha)$ classes for $k \geq 2$. To specify an $\mathbf{NC}^k(\alpha)$ circuit, we need to express the condition that $\wedge$ and $\vee$ gates have fanin 2. We use the following formula $Fanin2'(w, d, E, G)$ to express this, see also (4.19):

$$\forall z < d, \forall x < w$$
$$\left((G)^{\langle z,x \rangle} \neq \text{``}\alpha\text{''} \wedge (G)^{\langle z,x \rangle} \neq \text{``}\neg\text{''} \right) \supset Arity(z, x, 2, E).$$

Moreover, the nested depth of oracle gates in circuit (w, d, E, G) needs to be bounded separately from the circuit depth d. We use a formula

$$ODepth_k(w, d, E, G, D),$$

which states that this nested depth is bounded by $|w|^k$. Here the extra string variable D is to compute the nested depth of oracle gate: D is viewed as a sequence, where $(D)^{\langle z,x \rangle}$ is the oracle depth of gate (z, x). (Recall that the gates $(0, x)$ are input gates.) The sequence is computed inductively, starting with the input gates. An explicit formulation is rather straightforward but tedious, so we omit the details here. Note that we can use $\mathbf{AC}^0$ number functions such as $|x|$ and max which returns the maximum element in a bounded sequence), because they can be eliminated, see Cook & Nguyen (2010), Lemma V.6.7.

DEFINITION 4.25 ($\mathbf{VNC}^k(\alpha)$). *For $k \geq 2$, $\mathbf{VNC}^k(\alpha)$ is the theory over $\mathcal{L}_A^2(\alpha)$ and is axiomatized by $\mathbf{V}^0$ and the axiom*

(4.26) $[Proper(w, d, E) \wedge Fanin2'(w, |w|^k, E, G) \wedge$
$ODepth_{k-1}(w, d, E, G, D)] \supset \exists Q \exists Y \, \delta_{LOCV}^\alpha(w, |w|^k, E, G, I, Q, Y).$

PROPOSITION 4.27. *For $k \geq 2$, the problem of witnessing the quantifiers $\exists Q \exists Y$ in the axiom for $\mathbf{VNC}^k(\alpha)$ is complete for $\mathbf{NC}^k(\alpha)$ under $\mathbf{AC}^0$-many-one reductions.*

First we exhibit a problem complete under $\mathbf{AC}^0$reductions for $\mathbf{NC}^1(\alpha)$. Informally, this is the problem of evaluating a relativized sentence which is given using the *extended connection language* (Ruzzo 1981). More precisely, we consider encoding relativized sentences by tuples (a, G, I, J) in the following way. The sentence is viewed as a balanced binary tree as in the axiom *MFV* (4.10), but now each leaf $Y(x + a)$ can be an input bit (from I) or (the negation of) an α-gate that takes its input from J. In other words, the underlying circuit for the sentence has exactly one layer of oracle gates which take input directly from the input constants. More precisely, let

$$\delta(a, G, I, J, Y) \equiv \forall x < a, \ G(x + a) = \text{``}\alpha\text{''} \supset (Y(x + a) \leftrightarrow \alpha(J^{[x]})) \ \wedge$$
$$G(x + a) = \text{``}\neg\alpha\text{''} \supset (Y(x + a) \leftrightarrow \neg\alpha(J^{[x]})) \ \wedge$$
$$G(x + a) = \text{``const''} \supset (Y(x + a) \leftrightarrow I(x)) \ \wedge$$
$$0 < x \ \supset \ Y(x) \leftrightarrow$$
$$\big((G(x) \wedge Y(2x) \wedge Y(2x + 1)) \vee$$
$$(\neg G(x) \wedge (Y(2x) \vee Y(2x + 1)))\big) \, .$$

In the next result, we emphasize that the $\mathbf{AC}^0$reductions referred to are the 'Turing' reductions given in Definition 4.5.

LEMMA 4.28. *The relation given by the formula $\exists Y (\delta(a, G, I, J, Y) \wedge Y(1))$ is $\mathbf{AC}^0$-complete for $\mathbf{NC}^1(\alpha)$.*

PROOF. For the hardness direction, we note that the circuits solving a problem in $\mathbf{NC}^1(\alpha)$ have oracle nested depth bounded by some constant d. Hence such a circuit can be simulated by d circuits of the form described above, forming d layers. The layers can be evaluated by d successive queries to the relation in the lemma, where in each layer except the first, the constant inputs I and the oracle inputs J are determined by the gate values in the previous layer.

For membership in $\mathbf{NC}^1(\alpha)$, observe that we can evaluate the first layer of the circuit by an $\mathbf{AC}^0(\alpha)$ circuit (see also the proof of Proposition 4.23). Once this has been done, the remaining task is to evaluate a nonrelativized, balanced, monotone Boolean sentence, which can be done by an $\mathbf{NC}^1$ circuit.   □

PROOF (Proof outline of Proposition 4.27).   The hardness direction is proved as for Proposition 4.23: We assume that by definition an $\mathbf{NC}^k(\alpha)$circuit must satisfy the hypotheses of the axiom (4.26).

Now we argue that the problem actually belongs to $\mathbf{NC}^k(\alpha)$. Consider the case $k = 2$; other cases are similar. First, the given problem reduces to the following restriction of it, called P, where the layers in the given circuit are grouped together to form $|w|$ many blocks $B_1, B_2, \ldots, B_{|w|}$, where each block B_i has exactly $|w|$ layers and w outputs. Furthermore, each block is an $\mathbf{NC}^1(\alpha)$ circuit (with multiple outputs) such that all α-gates appear in the first layer. Moreover, these $\mathbf{NC}^1(\alpha)$ circuits are presented using the extended connection language. The reduction can be done by uniform circuits of polynomial size, $\log \log n$ depth and unbounded fanin, where n is the length of the input to our original problem.

It remains to show that the new problem P is solvable by a uniform family of $\mathbf{NC}^2(\alpha)$ circuits. Note that the input now can be viewed as the sequence

$$B_1, B_2, \ldots, B_{|w|},$$

where each B_i consists of w single-output $\mathbf{NC}^1(\alpha)$ circuits

$$B_{i,1}, B_{i,2}, \ldots, B_{i,w}.$$

Here each $B_{i,j}$ is an $\mathbf{NC}^1(\alpha)$ circuit where all α-gates are on the first layer. Lemma 4.28 above shows that each $B_{i,j}$ can be evaluated by an $\mathbf{NC}^1(\alpha)$ circuit $C_{i,j}$. As a result, the circuits for solving P are obtained by arranging $C_{i,j}$ appropriately.   □

The next theorem is proved in the same way as Theorem 4.24, using Proposition 4.27.

THEOREM 4.29. *For $k \geq 2$, the functions in $\mathbf{FNC}^k(\alpha)$ are precisely the $\Sigma_1^B(\alpha)$-definable functions of $\mathbf{VNC}^k(\alpha)$.*

Now we can apply the separations of the relativized classes obtained in Section 3 to prove separations of the corresponding theories.

COROLLARY 4.30. $\mathbf{VL}(\alpha) \subsetneq \mathbf{VAC}^1(\alpha)$, *and for $k \geq 1$:*

$$\mathbf{VAC}^k(\alpha) \subseteq \mathbf{VNC}^{k+1}(\alpha) \subsetneq \mathbf{VAC}^{k+1}(\alpha).$$

PROOF.   The first inclusion follows from Proposition 4.18, and the fact that the axiom $PATH(\alpha)$ is implied by the axiom for $\mathbf{VAC}^1(\alpha)$. The remaining inclusions are easy to check, so it suffices to show the strictness of the strict inclusions. By Theorems 4.15, 4.24 and 4.29, we know that the $\Sigma_1^B(\alpha)$-definable functions in each theory are those in the corresponding complexity class. By Corollary 3.9, we know that the inclusions of the corresponding complexity classes are strict, where indicated in the statement of the corollary. $\square$

5. Conclusion

The relativized class $\mathbf{AC}^k(\alpha)$, $k \geq 0$, has an obvious definition: treat an oracle gate $\alpha(x_1, \ldots, x_n)$ in the same way as $\wedge$ and $\vee$ gates. However, definitions of the relativized versions of the classes $\mathbf{NC}^k$, $\mathbf{L}$ and $\mathbf{NL}$ are not so obvious. Here we give new definitions for these classes that preserve many of the properties of the unrelativized classes, namely class inclusions, Savitch's Theorem and the Immerman-Szelepcsényi Theorem. However, there is a weakness in our definition of $\mathrm{cs}\mathbf{NL}(\alpha)$ (relativized $\mathbf{NL}$), namely the corresponding function class may not be closed under composition (all other function classes are so closed). A possible way out is to define relativized $\mathbf{NL}$ to be $\mathbf{AC}^0(\mathsf{STCONN}, \alpha)$ (see Theorem Theorem 2.9 part (ii) and its proof). This class has nice closure properties and satisfies the expected inclusions with other relativized classes. It also has a natural associated relativized theory, namely $\mathbf{VNL}'(\alpha)$ (see Definition Definition 4.14). But we do not

know how to define $\mathbf{AC}^0(\mathsf{STCONN}, \alpha)$ in terms of nondeterministic logspace oracle Turing machines. We leave this conundrum as an open problem.

We note that the first author has carried out in Aehlig (2010) a detailed study of propositional versions of our relativized theories.

Acknowledgements

References

KLAUS AEHLIG (2010). *Parallel Time and Proof Complexity*. Habilitation, Ludwig-Maximilians-University, Munich.

KLAUS AEHLIG, STEPHEN COOK & PHUONG NGUYEN (2007). Relativizing Small Complexity Classes and their Theories. In *16th EACSL Annual Conference on Computer Science and Logic*, 374–388. Springer. LNCS 4646.

DAVID A. MIX BARRINGTON, NEIL IMMERMAN & HOWARD STRAUBING (1990). On Uniformity within $\mathbf{NC}^1$. *Journal of Computer and System Sciences* **41**(3), 274–306.

JONATHAN BUSS (1986). Relativized Alternation. In *Proceedings Structure in Complexity Theory Conference*, 66–76. Springer.

SAMUEL BUSS (1987). The Boolean formula value problem is in **Alogtime**. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, 123–131.

STEPHEN COOK (1985). A Taxonomy of Problems with Fast Parallel Algorithms. *Information and Control* **64(1–3)**, 2–22.

STEPHEN COOK & PHUONG NGUYEN (2010). *Logical Foundations of Proof Complexity*. ASL Perspectives in Logic Series. Cambridge University Press.

NEIL IMMERMAN (1999). *Descriptive Complexity*. Springer.

Jan Krajíček (1995). *Bounded Arithmetic, Propositional Logic, and Complexity Theory.* Cambridge University Press.

Richard Ladner & Nancy Lynch (1976). Relativization of Questions About Log Space Computability. *Mathematical Systems Theory* **10**, 19–32.

Phuong Nguyen & Stephen Cook (2006). Theories for **TC**^{0} and Other Small Complexity Classes. *Logical Methods in Computer Science* **2**(1), 1–40.

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

Walter Ruzzo (1981). On Uniform Circuit Complexity. *Journal of Computer and System Sciences* **22**, 365–383.

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

Istvan Simon (1977). *On some subrecursive reducibilities.* Ph.D. thesis, Stanford University.

Gaisi Takeuti (1995). Separations of Theories in Weak Bounded Arithmetic. *Annals of Pure and Applied Logic* **71**, 47–67.

Christopher Wilson (1987). Relativized **NC**. *Mathematical Systems Theory* **20**, 13–29.

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

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

KLAUS AEHLIG
Mathematisches Institut
Ludwig-Maximilians-Universität
München
80333 München, Germany
aehlig@math.lmu.de

STEPHEN COOK
Department of Computer Science
University of Toronto
Toronto, ON M5S 3G4, Canada
sacook@cs.toronto.edu

PHUONG NGUYEN
Department of Computer Science
University of Toronto
Toronto, ON M5S 3G4, Canada
phuong.the.nguyen@gmail.com