# EXPRESSIBILITY AS A COMPLEXITY MEASURE: RESULTS AND DIRECTIONS

Neil Immerman
YALEU/DCS TR 538
April, 1987

# Expressibility as a Complexity Measure: Results and Directions

Neil Immerman*

*Computer Science Department*
*Yale University*
*New Haven, CT 06520*

April 10, 1987

## 1 Introduction

Given a property, $S$, one can discuss the computational complexity of checking whether or not an input satisfies $S$. One can also ask, "What is the complexity of *expressing* the property $S$?" It is natural that these two questions are related. However, it is startling how closely tied they are when the second question refers to expressing the property in first-order logic. In this article we survey some work relating first-order expressibility to computational complexity, and present a number of open problems. Most complexity theory is done from the point of view of Turing machines, or boolean circuits. We believe that first-order logic provides a valuable different point of view, adding fresh insights to well studied problems.

## 2 Background and History

Our first ideas concerning expressibility were inspired by the following result of Fagin:

**Theorem 2.1** *[13] NP = (2nd O, ∃).*

*Research supported by NSF Grant DCR-8603346.

This theorem says that if we consider inputs as logical structures, then the NP properties are exactly those properties expressible as second order existential sentences. A typical example is the set of three colorable graphs. The corresponding sentence is $(\exists R \exists Y \exists B)\varphi(R, Y, B)$ where R,Y, and B, are new monadic relations and $\varphi$ is a first-order sentence saying that R,Y, and B, form a valid three coloring of the input graph.

Theorem 2.1 gives a beautiful characterization of NP. As a corollary, Stockmeyer observed that a problem is in the polynomial time hierarchy if and only if it is expressible in second order logic:

**Corollary 2.2** *[36] PH = (2nd O)* .

## 2.1 First-Order Logic

In order to model computations in a more detailed manner, we chose to consider the relations between the complexity of properties and their expressibility in first-order logic. We will describe these close relationships in the rest of this paper. First, it is necessary to make some precise definitions. The reader is refered to [12] for more information about first-order logic.

A *vocabulary* $\tau = \langle \underline{R}_1^{a_1} ... \underline{R}_k^{a_k}, \underline{c}_1 ... \underline{c}_r \rangle$ is a tuple of relation symbols and constant symbols. $\underline{R}_i^{a_i}$ is a relation symbol of arity $a_i$. In the sequel we will usually omit the superscripts and the underlines to improve readability. A finite *structure* with vocabulary $\tau$ is a tuple, $\mathcal{A} = \langle \{0, 1, \ldots, n-1\}, R_1^{\mathcal{A}} ... R_k^{\mathcal{A}}, c_1^{\mathcal{A}} ... c_r^{\mathcal{A}} \rangle$, consisting of a universe $|\mathcal{A}| = n = \{0, \ldots, n-1\}$ and relations $R_1^{\mathcal{A}} ... R_k^{\mathcal{A}}$ of arities $a_1, \ldots, a_k$ on $|\mathcal{A}|$ corresponding to the relation symbols $\underline{R}_1^{a_1} ... \underline{R}_k^{a_k}$ of $\tau$, and constants $c_1^{\mathcal{A}} ... c_r^{\mathcal{A}}$ from $|\mathcal{A}|$ corresponding to the constant symbols $\underline{c}_1 ... \underline{c}_r$ from $\tau$.

For example, if $\tau_0 = \langle \underline{E}^2 \rangle$ consists of a single binary relation symbol then a structure $G = \langle \{0...n-1\}, E \rangle$ with vocabulary $\tau_0$ is a graph on n vertices. Similarly if $\tau_1 = \langle \underline{M}^1 \rangle$ consists of a single monadic relation symbol then a structure $S = \langle \{0...n-1\}, M \rangle$ with vocabulary $\tau_1$ is a binary string of length n.

Let the symbol '$\leq$' denote the usual ordering on the natural numbers. We will include $\leq$ as a logical relation in our first-order languages. This seems necessary inorder to simulate machines whose inputs are structures given in some order. For convenience we also include the successor relation, $s(x, y)$, and the constant symbols $\underline{0}, \underline{max}$ refering to the first and last elements of the structure respectively. For technical reasons we also include the logical predicate BIT, where $\mathrm{BIT}(x, y)$ holds iff the xth bit in the binary expansion

of y is a one.[1]

We now define the *first-order language* $\mathcal{L}(\tau)$ to be the set of formulas built up from the relation and constant symbols of $\tau$ and the logical relation symbols and constant symbols: $=, \leq, s, \mathrm{BIT}, \underline{0}, \underline{\max}$, using logical connectives: $\wedge, \vee, \neg$, variables: $x, y, z, ...$, and quantifiers: $\forall, \exists$.

We will think of a *problem* as a set of structures of some vocabulary $\tau$. Of course it suffices to only consider problems on binary strings, but it is more interesting to be able to talk about other vocabularies, e.g. graph problems, as well. Define FO to be the set of all first-order expressible problems.

We will see in section 3 that FO is a nice, uniform version of $AC^0$. However, in order to consider more powerful complexity classes it is useful to let the size of our first-order sentences grow with the size of the input structures:

## 2.2 The Complexity Classes VAR&SZ

Let $|\varphi|$ denote the *size* – i.e. number of symbols – of the formula $\varphi$. The number of distinct variables is also an important resource for first-order sentences. Variables in a first-order sentence may be requantified: the quantifier $(\forall x_i)$ binds only the free occurrences of $x_i$ in its scope. For any vocabulary $\tau$ let $\mathcal{L}_k(\tau)$ be the restriction of $\mathcal{L}(\tau)$ to formulas in which no variables besides $x_1, x_2, \ldots, x_k$ occur. We now define the complexity classes VAR&SZ$[v(n), z(n)]$ to be the set of problems expressible by a uniform sequence of first-order sentences with $v(n)$ variables and size $z(n)$.

**Definition 2.3** *[22] A set $C$ of structures of vocabulary $\tau$ is a member of VAR&SZ$[v(n), z(n)]$ iff there exists a uniform[2] sequence of sentences $\{\varphi_n\}_{n \in \mathbf{N}}$ such that*

1. *For all structures $G$ of vocabulary $\tau$ with $|G| \leq n$,*

$$G \in C \Leftrightarrow G \models \varphi_n .$$

2. *$|\varphi_n| = O[z(n)]$, and $\varphi_n \in \mathcal{L}_{v(n)}(\tau)$.*

[1] Of course some of these logical relations are redundant. We include all of them to make our life easier. The use of BIT will be explained in Section 3, and the use of the constant symbols and $s$ will be explained in Section 4.

[2] A sufficient uniformity condition is that the map from $n$ to $\varphi_n$ can be generated in DSPACE$[v(n) \log n]$ and simultaneously DTIME$[z(n) \log z(n)]$. Below we develop a simple, syntactic uniformity condition when $v(n)$ is constant.

The next theorem characterizes the relationship between the VAR&SZ complexity classes and simultaneous space and time on alternating Turing machines [7]. Each variable in our sentences ranges between 0 and $n-1$ and thus consists of $\log n$ bits. Thus ASPACE$[s(n)]$ corresponds to $O[s(n)/\log n]$ variables. Similarly an existential (resp. universal) quantifier can simulate $\log n$ existential (resp. universal) steps in a row on an alternating machine. Thus the essential difference between the power of the two models is that the alternating machines may alternate at every step, but the formulas only every $\log n$ steps. Define ASPACE&TIME&ALT$[s(n), t(n), a(n)]$ to be the complexity class of problems accepted by an alternating Turing machine simultaneously using space $s(n)$, time $t(n)$, and making a total of $O[a(n)]$ alternations. Then we have,

**Theorem 2.4** *[22, Theorem B.4] For $s(n) \geq \log n$ and $t(n) \leq 2^{s(n)}$,*

$$ASPACE\&TIME\&ALT[s(n), t(n), t(n)/\log n] = VAR\&SZ[O[s(n)/\log n], t(n)/\log n] \quad .$$

Of course one obtains many corollaries from Theorem 2.4 concerning the relations between expressibility and deterministic complexity. One particularly interesting observation, though, is that VAR&SZ$[v(n), z(n)]$ makes sense for $z(n) > n^{v(n)}$. In particular when we restrict attention to polynomial space we find,

**Theorem 2.5** *[22] For $t(n) \geq n$,*

$$\bigcup_{c,k=1}^{\infty} DSPACE\&TIME[n^c, t(n)^k] = \bigcup_{c,k=1}^{\infty} VAR\&SZ[c, t(n)^k] \, .$$

In particular it follows that,

**Corollary 2.6** *[22]*

$$P = \bigcup_{c,k=1}^{\infty} VAR\&SZ[c, n^k]$$

$$PSPACE = \bigcup_{c,k=1}^{\infty} VAR\&SZ[c, 2^{n^k}]$$

The above corollary already gives a hint of the dramatic view of complexity that arises when we restrict our attention to constantly many variables. In particular, the uniformity condition for the constantly many variables case is purely syntactic, thus giving a uniform characterization of most interesting complexity classes using natural notions coming entirely from logic. We will concentrate on this picture in the next section.

**Open Problem 1** *The following bounds can be derived from the proof of Theorem 2.5 in [22],*

$$\begin{aligned} DTIME[n^k] &\subseteq VAR\&SZ[k+3, n^k] \\ &\subseteq DTIME[n^{2k+4}] \end{aligned}$$

*We believe these can be improved and it would be of significant interest to obtain tight bounds, cf. [17,10].*

# 3 Iterating First-Order Sentences

In [22] we observed that a good uniformity condition for the complexity classes VAR&SZ$[O[1], t(n)]$ is that the sentences $\varphi_n$ consist of a "single sentence iterated $t(n)$ times." We will use the notation $FO[t(n)]$ to refer to these complexity classes. We now make this notion of iterating a sentence precise.

Let $x$ be a variable and $M$ a quantifier free formula. We will use the notation $(\forall x.M)\psi$ – read, "for all $x$ such that $M$, $\psi$," – to abbreviate $(\forall x)(M \rightarrow \psi)$. Similarly we wil write $(\exists x.M)\psi$ – read, "there exists an $x$ such that $M$, $\psi$," – to abbreviate $(\exists x)(M \wedge \psi)$. We will call the expressions $(\forall x.M)$ and $(\exists x.M)$ *restricted quantifiers.* Let a *quantifier block* be a finite sequence of restricted quantifers: $QB = (Q_1 x_1.M_1)\ldots(Q_k x_k.M_k)$. We will use the notation $[QB]^t$ to denote the quantifier block $QB$ repeated $t$ times. Note that for any quantifier free formulas $M_0, M_1, \ldots, M_k \in \mathcal{L}(\tau)$, and any $t \in \mathbf{N}$, the expression $[QB]^t M_0$ is a well formed formula in $\mathcal{L}(\tau)$.

**Definition 3.1** *A set $C$ of structures of vocabulary $\tau$ is a member of $FO[t(n)]$ iff there exists a quantifier block $QB$ and a quantifier free formula $M_0$ from $\mathcal{L}(\tau)$, such that if we let $\varphi_n = [QB]^{t(n)} M_0$, for $n = 1, 2, \ldots$, then for all structures $G$ of vocabulary $\tau$ with $|G| \leq n$,*

$$G \in C \Leftrightarrow G \models \varphi_n .$$

A more traditional way to iterate formulas is by making inductive definitions, [32,23]. Let IND$[t(n)]$ be the set of problems expressible as a uniform induction which requires depth of recursion at most $t(n)$ for structures of size $n$. The following lemma relates IND$[t(n]$ to $FO[t(n)]$.

**Lemma 3.2** *[25]*

1. *$IND[t(n)] \subseteq FO[t(n)]$ for all $t(n)$. In particular, a property in $IND[t(n)]$ is expressible as a $FO[t(n)]$ property in which $M_0 \equiv (false)$.*
2. *$IND[t(n)] = FO[t(n)]$ if $t(n)$ is time constructible by an $ASPACE[\log n]$ Turing machine.*

**Example 3.3** *As an example we show how to go from a $\log n$ depth inductive definition of the transitive closure of a graph to a $FO[\log n]$ definition of transitive closure.*

*Let $E$ be the edge predicate for a graph $G$ with $n$ vertices. We can inductively define $E^*$ the reflexive,transitive closure of $G$ as follows:*

$$E^*(x,y) \equiv x = y \vee E(x,y) \vee (\exists z)(E^*(x,z) \wedge E^*(z,y)) \ .$$

*Let $P_n(x,y)$ mean that there is a path of length at most $n$ from $x$ to $y$. Then we can rewrite the above definition of $E^*$ as:*

$$P_n(x,y) \equiv x = y \vee E(x,y) \vee (\exists z)(P_{n/2}(x,z) \wedge P_{n/2}(z,y)) \ .$$

*This can be rewritten:*

$$P_n(x,y) \equiv (\forall z.M_1)(\exists z)(P_{n/2}(x,z) \wedge P_{n/2}(z,y))$$

*where* $M_1 \equiv \neg(x = y \vee E(x,y))$. *Next,*

$$P_n(x,y) \equiv (\forall z.M_1)(\exists z)(\forall uv.M_2)(P_{n/2}(u,v))$$

*where* $M_2 \equiv (u = x \wedge v = z) \vee (u = z \wedge v = y)$. *Now,*

$$P_n(x,y) \equiv (\forall z.M_1)(\exists z)(\forall uv.M_2)(\forall xy.M_3)(P_{n/2}(x,y))$$

*where* $M_3 \equiv (x = u \wedge y = v)$. *Thus,*

$$P_n(x,y) \equiv [QB]^{\lceil \log n \rceil}(P_1(x,y))$$

*where* $QB = (\forall z.M_1)(\exists z)(\forall uv.M_2)(\forall xy.M_3)$. *Note that*

$$P_1(x,y) \equiv [QB](false)$$

*It follows that*

$$P_n(x,y) \equiv [QB]^{\lceil 1+\log n \rceil}(false)$$

*and thus $E^* \in FO[\log n]$ as claimed.*

## 3.1 FO and Parallel Complexity

We now describe some results relating parallel complexity to first-order expressibility. All the material in this subsection may be found in more detail in [25].

Let a CRAM be essentially a Concurrent Read, Concurent Write Parallel Random Access Machine (CRCW PRAM) [37,25]. These well studied, idealized parallel computers consist of a certain number of processors each of which may access any word of the unbounded global memory at each synchronous step. The convention is that if several processors try to write into the same location at the same time step then the lowest numbered processor will succeed. The definition of the CRAM differs from the standard definition of the CRCW PRAM in [37] only in that the CRAM may shift a word of memory by $\log n$ bits in unit time. Thus for parallel time greater than or equal to $\log n$ there is no distinction between the two models.

Define CRAM$[t(n)]$ to be the complexity class of problems solvable by a CRAM using polynomially many processors and parallel time $t(n)$. We have the following fairly strong connection between parallel time and first-order expressibility:

**Theorem 3.4** *[25] For all polynomially bounded $t(n)$,*

$$CRAM[t(n)] \;=\; FO[t(n)]\,.$$

**Open Problem 2** *The proof of Theorem 3.4 in [25] gives the following bounds for translating from CRAM to FO. Let $CRAM\&PROC[t(n), p(n)]$ be the complexity class $CRAM[t(n)]$ restricted to machines using at most $O[p(n)]$ processors. Then for $t(n) \leq n$ we have,*

$$\begin{aligned} CRAM\&PROC[t(n), n^k] \;&\subseteq\; VAR\&SZ[2k+4, t(n)] \\ &\subseteq\; CRAM\&PROC[t(n), n^{2k+4}] \end{aligned}$$

*We feel that these bounds can and should be significantly improved. We would especially like to know if the processor blow-up can be reduced to a linear factor, perhaps by allowing a small increase in $t(n)$.*

For $t(n) \geq \log n$, Theorem 3.4 may also be obtained as a corollary of Theorem 2.4 together with a result of Ruzzo and Tompa relating CRAMs to alternating Turing machines [37]. In order to prove the theorem for $t(n) < \log n$ we were forced to modify the models slightly, adding the $\log n$ shift operation to the CRAMs, and adding BIT as a new logical relation to

our first-order language. We believe that the naturalness of Theorem 3.4 justifies these modifications.

We also obtain corollaries relating FO[$t(n)$] to the NC and AC circuit classes. Let $NC^i$ (resp. $AC^i$) be the set of problems recognizable by a uniform sequence of polynomial size, bounded fan-in (resp. unbounded fan-in) boolean circuits. Let $NC = AC = \bigcup_i NC^i$. Ruzzo [33] characterized these uniform circuit classes in terms of alternating Turing machines. See [?] for an overview of these classes.

**Theorem 3.5** *[33] For $i \geq 1$,*

$$\begin{aligned} NC^i &= ASPACE\&TIME[\log n, \log^i n] \\ AC^i &= ASPACE\&ALT[\log n, \log^i n] \end{aligned}$$

**Corollary 3.6** *[25] For $i \geq 1$, $AC^i = FO[\log^i n]$.*

In [8] it is shown that in the non-uniform case, the obvious bounds nonuniform $NC^i \subseteq$ nonuniform $AC^i$ can be improved to

$$\text{nonuniform}NC^i \subseteq \text{nonuniform AC-depth}[log^i n / \log\log n] \ .$$

When $i = 1$ this bound is optimal because nonuniform AC-depth$[logn / \log\log n]$ is necessary and sufficient for Parity [20]. Furthermore, the same bound holds in the uniform case:

**Theorem 3.7** *[25] For $i \geq 1$, $NC^i \subseteq FO[\log^i n / \log\log n]$ .*

Consider the following characterizations of nonuniform $AC^0$:

**Theorem 3.8** *[37] nonuniform $AC^0$ = nonuniform $CRAM[1]$.*

**Theorem 3.9** *[22] nonuniform $AC^0$ = nonuniform $FO[1]$.*

The above two results, together with Theorems 3.4 and 3.7, justify the following

**Definition 3.10** *Let (uniform) $AC^0 \stackrel{\text{def}}{=} FO[1] = CRAM[1]$ .*

**Open Problem 3** *Evaluate this definition for (uniform) $AC^0$. In particular, is it too restrictive?*

Theorem 3.7 makes us want to prove an $\Omega[\log n]$ lower bound on transitive closure. This is Open Problem 10 which we discuss in Section 5.

**Open Problem 4** *Steve Homer recently suggested the following problem: Given a function $t(n)$, derive a (simple) quantifier block, QB, and formula $M_0$ such that the property $[QB]^{t(n)}M_0$ is complete for the class FO$[t(n)]$. In particular, can the same QB work for more than one $t(n)$? How about for all $t(n)$?*

Figure 1 summarizes many of the known relationships between complexity classes and the FO$[t(n)]$ expressibility classes. We also include a few expressibility classes involving new operators, which we will explain in the next section.

**Open Problem 5** *Figure 1 shows that as $t(n)$ ranges from $\log n/\log\log n$ through $2^{n^i}$, FO$[t(n)]$ ranges through most of the interesting complexity classes. Almost any hierarchy theorem in this interval would settle an important open problem. However, this is not a sufficient reason to be scared off. There is still plenty of room for partial results concerning the trade off between size and number of variables. (Two significant hierarchy results for FO$[1]$ and FO$[\log n/\log\log n]$ are [34] and [20], respectively.)*

# 4 Adding Operators to First-Order Logic

At first glance the idea of iterating a first-order sentence in the definition of FO$[t(n)]$ may seem unusual. In this section we study an alternative approach: adding operators to first-order logic.

One likely place to begin is by adding the least fixed point operator (LFP). LFP is a standard way to formalize inductive definitions [32,6,23]. Suppose that $\varphi(R^k, x_1, \ldots, x_k)$ is a formula in which the new k-ary relation symbol $R^k$ occurs only positively, i.e. within an even number of negation symbols. Then for any structure $\mathcal{A}$, $\varphi$ defines a monotone map $\varphi_{\mathcal{A}}$ from k-ary relations on $\mathcal{A}$ to k-ary relations on $\mathcal{A}$:

$$\varphi_{\mathcal{A}} : S \mapsto \{\langle a_1, \ldots, a_k\rangle \in |\mathcal{A}|^k \mid \mathcal{A} \models \varphi(S, a_1, \ldots, a_k)\}$$

Since $\varphi_{\mathcal{A}}$ is monotone, it has a least fixed point, which may be calculated by starting with the empty relation and applying $\varphi_{\mathcal{A}}$ to it repeatedly until a fixed point is reached – after at most $n^k$ applications where $n = |\mathcal{A}|$.

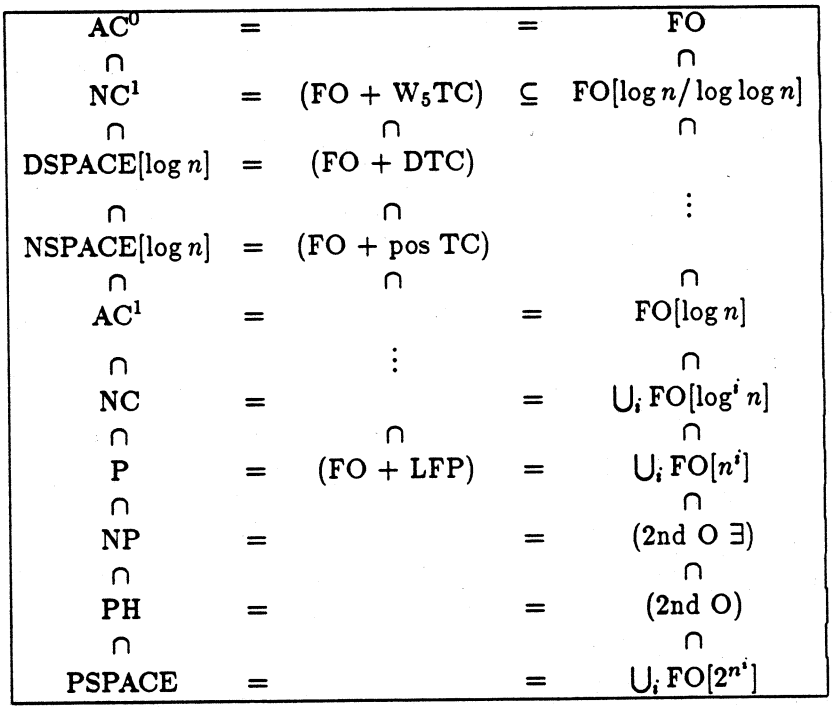


Figure 1: Complexity versus Expressibility

We will adopt the notation $\mathrm{LFP}[\lambda R^k, x_1, \ldots, x_k\, \varphi]$ to denote the least fixed point of $\varphi$. Let (FO + LFP) be the closure of FO under LFP applied to positive formulas. The following theorem follows immediately from Corollary 2.6 and Lemma 3.2.

**Theorem 4.1** *[22,41] P = (FO + LFP) .*

**Example 4.2** *Continuing Example 3.3, we would formalize the inductive definition of $E^*$ as follows: Let*

$$\alpha(R, x, y) \equiv x = y \vee E(x, y) \vee (\exists z)(R(x, z) \wedge R(z, y)) .$$

*Then $E^* = LFP[\lambda R^2, x, y\, \alpha]$.*

It follows from Lemma 3.2 that LFP may be thought of simply as an iteration operator. See [5] for an elaboration of this point of view as well as a study of the power of LFP in the more subtle case when the ordering relation $\leq$ is not present. See also [23,19].

**Open Problem 6** *[5] Sam Buss and I show in [5] that even when ordering is not present, (FO + LFP) = $\bigcup_\alpha FO[\alpha]$, where $\alpha$ ranges over the closure ordinals of positive formulas. (The closure ordinal of a formula is the depth of recursion of the induction defined by the formula.) An intriguing question is then, what closure ordinals can occur in various sets of unordered structures? A highly related question was posed by Steven Lindell [29]: For which sets of structues, $\Sigma$, besides those containing ordering, does (FO + LFP) = P?*

Another very natural kind of operator to add is the transitive closure operator, TC.

**Definition 4.3** *For any formula $\varphi$ in which the free variables $x_1, \ldots, x_k, x'_1, \ldots, x'_k$ may occur, let the notation $TC[\lambda \overline{x}, \overline{x'}\, \varphi]$ denote the reflexive, transitive closure of the binary relation $\varphi(\overline{x}, \overline{x'})$. Let (FO + TC) be the closure of FO under this formation rule for TC. Let (FO + pos TC) be that part of (FO + TC) in which there is no occurrence of TC within any negation symbols.*

**Theorem 4.4** *[24] NSPACE[$\log n$] = (FO + pos TC).*

In [24] we introduced and studies several transitive closure operators. Here we mention one other: the deterministic transitve closure, DTC.

**Definition 4.5** *For any formula $\varphi(x_1, \ldots, x_k, x'_1, \ldots, x'_k)$, let the deterministic version of $\varphi$ be given as follows:*

$$\varphi_d(\overline{x}, \overline{x'}) \equiv \varphi(\overline{x}, \overline{x'}) \wedge \forall \overline{y}(\varphi(\overline{x}, \overline{y}) \rightarrow \overline{y} = \overline{x'}) .$$

*That is, $\varphi_d$ has at most one 'edge' leaving any 'vertex'. Define $DTC[\lambda \overline{x}, \overline{x'}\, \varphi]$ to be an abbreviation for $TC[\lambda \overline{x}, \overline{x'}\, \varphi_d]$*

**Theorem 4.6** *[24] $DSPACE[\log n] = (FO + DTC)$.*

Let $\Sigma_k^L$ be the class of problems recognizable by an ASPACE[$\log n$] machine making only k-1 alternations, and beginning in an existential state. It was recently shown that the Logspace Hierarcy $\bigcup_k \Sigma_k^L$ collapses at $\Sigma_2^L$, [38]. There is a corresponding (FO + TC) Hierarchy in which one alternates occurrences of TC and negation, e.g. (FO + pos TC) = $\Sigma_1^{TC}$. We know the following:

**Theorem 4.7** *[24]*

$$\begin{aligned} DSPACE[\log n] &= (FO + DTC) \\ &\subseteq NSPACE[\log n] = \Sigma_1^{TC} \\ &\subseteq \bigcup_k \Sigma_k^L \quad (= \Sigma_2^L [38]) \\ &\subseteq \Sigma_2^{TC} \\ &\subseteq (FO + TC) = \bigcup_k \Sigma_k^{TC} \\ &\subseteq FO[\log n] \\ &\subseteq (FO + LFP) = P \end{aligned}$$

**Open Problem 7** *Theorem 4.7 leaves an open question for each of the six '$\subseteq$' symbols, namely, "Is it '$\subset$', or is it '='?" Of special interest – because these seem tractable – are the questions: "Does the (FO + TC) Hierarchy collapse?" and if so, "Is (FO + TC) = $FO[\log n]$?"*

## 4.1 First-Order Translations and Projection Translations

We were surprised to find that all the operators we have considered satisfy a very restrictive normal form theorem. For example,

**Theorem 4.8** *[24] Let $\Phi$ be the operator TC (resp. DTC), and let $\alpha \in$ (FO + pos TC) (resp. (FO + DTC)). Then there exists a constant $k$ and a first-order formula $\beta(R^k, u_1, \ldots u_k, u'_1, \ldots u'_k)$ such that $\alpha$ is equivalent to*

$$\Phi[\lambda R^k, u_1, \ldots u_k, u'_1, \ldots u'_k \beta](\overline{0}, \overline{\max}) .$$

*where $\overline{0}, \overline{\max}$ are the constant symbols $0, \max$ repeated $k$ times.*

Theorem 4.8 says that only one application of the operator is needed in (FO + DTC) and (FO + pos TC). It follows that versions of these operators are complete for these classes via first-order translations, which we now define.

**Definition 4.9** *cf. [12,24] Let $\tau_1$ and $\tau_2$ be vocabularies where $\tau_2 = \langle \underline{R}_1^{a_1} \ldots \underline{R}_r^{a_r} \rangle$. Let $k$ be a constant. An* interpretation, *$\sigma$, of $\mathcal{L}(\tau_2)$ in $\mathcal{L}(\tau_1)$ is a sequence of $r$ formulas from $\mathcal{L}(\tau_1)$: $\Sigma_i(x_{11}, \ldots, x_{a_i k})$ $i = 1 \ldots r$, where each $\Sigma_i$ has $k \cdot a_i$ free variables.*

Such an interpretation $\sigma$ translates any structure $\mathcal{A}$ of vocabulary $\tau_1$ to a structure $\sigma(\mathcal{A})$ of vocabulary $\tau_2$ as follows: $|\sigma(\mathcal{A})| = n^k$. Each element $t \in |\sigma(\mathcal{A})|$ is associated with a k-tuple, $t_1, \ldots, t_k$, from $|\mathcal{A}|$ via the lexicographical ordering of k-tuples. Thus, for example, $0_i^{\sigma(\mathcal{A})} = 0^{\mathcal{A}}$, $i = 1, \ldots, k$. Each relation $R_i \in \tau_2$ is defined in $\sigma(\mathcal{A})$ using the corresponding $\Sigma_i$:

$$R_i^{\sigma(\mathcal{A})} = \{\langle t^1, \ldots, t^{a_i} \rangle \mid \mathcal{A} \models \Sigma_i(t_1^1, t_2^1, \ldots, t_k^{a_i})\}$$

**Definition 4.10** *Given two problems: $S \subset STRUCT[\tau_1]$ and $T \subset STRUCT[\tau_2]$,* a first order translation *of $S$ to $T$ is an interpretation, $\sigma$, of $\mathcal{L}(\tau_2)$ in $\mathcal{L}(\tau_1)$ such that:*

$$G \in S \Leftrightarrow \sigma(G) \in T$$

Thus a first-order translation is a reduction between two problems given by a fixed r-tuple of first-order formulas. The following is immediate from Definitions 4.9, 4.10 and Theorem 4.8.

**Corollary 4.11** *[24] Let $GAP = \{G|TC[E](0, max)\}$ and let $1GAP = \{G|DTC[E](0, max)\}$. Then GAP is complete for (FO + pos TC) (i.e. $NSPACE[\log n]$) and 1GAP is complete for (FO + DTC) (i.e. $DSPACE[\log n]$), both via first-order translations.*

It follows from Theorem 3.4 and Definition 3.10 that the first-order translation is a uniform version of constant depth reducibility [8]. It would seem that the projections of Valiant [40], are even weaker.

**Definition 4.12** *[40] Let $S$ and $T$ be problems on boolean strings. $S$ is said to be a* projection *of $T$ if there exists a constant $k$, and, for each $n$, a map,*

$$\rho_n : \{x_1, \ldots, x_{n^k}\} \rightarrow \{y_1, \bar{y}_1, \ldots, y_n, \bar{y}_n\} \cup \{0, 1\}$$

*such that*

$$S = \{y_1 \ldots y_n \mid \rho_n(x_1), \ldots, \rho_n(x_{n^k}) \in T\}$$

Thus, $S$ is a projection of $T$ iff the circuits for $T$ also work for $S$ with polynomial blow-up, and a rewiring of the inputs.

A first-order structure may be coded as a binary string listing the truth value for each of its non-logical predicates. (For example, a graph is coded as its $n^2$ bit adjacency matrix.) Note that the input bits correspond to atomic occurrences of non-logical relations. We found that a stronger version of Theorem 4.8 holds:

**Theorem 4.13** *[24] Let $\Phi$ be the operator TC (resp. DTC), and let $\alpha \in$ (FO + pos TC) (resp. (FO + DTC)). Then there exists a constant $k$ and a first-order formula $\beta(R^k, \overline{u}, \overline{u'})$ such that $\alpha$ is equivalent to*

$$\Phi[\lambda R^k, u_1, \ldots u_k, u'_1, \ldots u'_k \beta](\overline{0}, \overline{\max}) .$$

*where $\beta$ is a quantifier free formula in disjunctive normal form:*

$$\beta \equiv \bigvee_{i=1}^{I} \gamma_i \wedge \delta_i \qquad \textit{where:}$$

1. *Each $\gamma_i$ is a conjunction of the logical atomic relations, $s, =$, and their negations.*
2. *Each $\delta_i$ is atomic or negated atomic.*
3. *For $i \neq j$, $\gamma_i$ and $\gamma_j$ are mutually exclusive.*

Call a first order translation $\sigma = \langle \Sigma_1, \ldots, \Sigma_r \rangle$ a *projection translation* if each of its formulas is in the form of $\beta$ in the above theorem. It is easy to see that,

**Corollary 4.14** *[24]*

1. *GAP is complete for NSPACE*[log $n$] *via projection translations.*
2. *1GAP is complete for DSPACE*[log $n$] *via projection translations.*

**Corollary 4.15** *[24] NP = DSPACE*[log $n$] *iff CLIQUE is a projection translation of 1GAP.*

**Remark 4.16** *The projection translation is a uniform version of Valiant's projection. The maps $\rho_n$ in the definition of projection are all given by a fixed quantifier free formula, $\beta$.*

**Open Problem 8** *One would solve an important problem by showing that CLIQUE is not a projection translation of 1GAP. It would be very worthwhile to study the projection translation, and more generally the first-order translation, in detail. In particular it would be of great interest to prove a lower bound on k for which a k-ary projection translation from CLIQUE to 1GAP exists, cf. [16].*

## 4.2 $NC^1$ and Bounded-Width Branching Programs

David Barrington recently showed that $NC^1$ is equal to the set of problems recognizable by polynomial-size, width 5 branching programs.

**Theorem 4.17** *[3]*

1. *$NC^1$ = Bounded-Width, Polynomial-Size Branching Programs.*
2. *The word problem for $S_5$ is complete for $NC^1$ via non-uniform $AC^0$ reductions.*

One can define a version of TC restricted to width 5 graphs, which we will call $W_5$TC. Similarly, one can define a corresponding version of GAP called $W_5$GAP. The following is then proved by checking that the non-uniform $AC^0$ reductions in Theorem 4.17 can in fact be made uniform. This adds futher support to Definition 3.10.

**Theorem 4.18** 1. *$NC^1$ = (FO + $W_5$TC)*

2. *The word problem for $S_5$ and $W_5$GAP are complete for $NC^1$ via first-order translations.*

**Open Problem 9** *Are the word problem for $S_5$ and $W_5$GAP complete for $NC^1$ via projection translations?*

# 5 Lower Bounds

When we began this line of research one of the most exciting directions seemed to be lower bounds. For example, to prove P$\neq$ $AC^1$ it suffices to show that some property in P cannot be expressed by sentences of length $O[\log n]$. Similarly, proving P $\neq$ NP is equivalent to showing that some second-order property is not in (FO + LFP). Of course even when we cast these problems in this new light, they are still very hard. In this section we review the little we can show about lower bounds and suggest some problems and future directions to be tackled.

## 5.1 Ehrenfeucht-Fraisse Pebble Games

One way to prove that a problem $S$ is not describable in a language $\mathcal{L}$ is to produce a pair of structures, $\mathcal{A} \in S, \mathcal{B} \notin S$, such that that $\mathcal{A}$ and $\mathcal{B}$ are $\mathcal{L}$ equivalent, i.e. agree on all sentences from $\mathcal{L}$. In [22] we showed how to modify the standard Ehrenfeucht-Fraisse games [11,14] to include pebbles.[3] The game $G(\mathcal{A}, \mathcal{B}, k, n)$ is played for $n$ moves with $k$ pairs of pebbles on structures $\mathcal{A}, \mathcal{B}$. There are two players: Player I is trying to show that $\mathcal{A} \ncong \mathcal{B}$ and Player II is trying to prevent Player I from succeeding. Let $\mathcal{L}_{k,m}$ be the language $\mathcal{L}_k$ restricted to sentences of quantifier rank at most $m$. (The quantifier rank is the maximum depth of nesting of quantifiers in a formula.) We have:

**Theorem 5.1** *[22] The following are equivalent:*

1. *Player II has a forced win in the game $G(\mathcal{A}, \mathcal{B}, k, m)$.*
2. *$\mathcal{A}$ and $\mathcal{B}$ are $\mathcal{L}_{k,m}$ equivalent.*

By Theorem 5.1 the above pebbling games provide a convenient tool for proving the $\mathcal{L}_{k,m}$ equivalence of structures, and thus proving VAR& SZ$[k, m]$ lower bounds. A problem arises however in that when the logical relation '$\leq$' is present, two structures that look the same are equal:

**Proposition 5.2** *[21,22]*

*IF $|\mathcal{A}| \leq n$, $\mathcal{L}$ includes '$\leq$' and $\mathcal{A}$ is $\mathcal{L}_{3,3+\log n}$ equivalent to $\mathcal{B}$*

*THEN $\mathcal{A} = \mathcal{B}$.*

[3] See also [4] for a similar modification.

Proposition 5.2 is a significant stumbling block to proving lower bounds greater than FO[$\log n$]. A straight forward Ehrenfeucht-Fraisse argument shows that $\Omega[\log n]$ quantifiers are necessary in Proposition 5.2 and also necessary to express deterministic transitive closure, even with ordering present (but without BIT):

**Theorem 5.3** *[21] If $t(n)$ is $o[\log n]$ then 1GAP $\notin$ FO(w.o. BIT)$[t(n)]$.*

**Open Problem 10** ***Extend Theorem 5.3 to include BIT. This would have the very interesting consequence that $NC^1 \neq DSPACE[\log n]$, cf. Theorem 3.7. We feel that this problem may be tractable and thus is worth considerable effort.***

## 5.2 Lower Bounds Without Ordering

In [21,22] we proved several lower bounds on VAR& SZ(w.o. $\leq$)$[v(n), z(n)]$. Here when we say, 'w.o. $\leq$,' we also exclude $s$ and BIT. Since the languages without ordering are not strong enough to simulate Turing machines it is difficult to interpret the meaning of the following three theorems. We believe, however, that they provide some intuitive support for the conjectures: NC $\neq$P, P$\neq$NP, and Graph Isomorphism $\notin$ P, respectively.

**Theorem 5.4** *[21] The problem AGAP is in P, but AGAP is not in VAR& SZ(w.o.$\leq$)$[t(n), t(n)]$ for $t(n) = o[2^{\sqrt{\log n}}]$.*

**Theorem 5.5**
1. *[22,10] For all $k$, "Has a Hamilton Circuit" $\notin$ VAR(w.o. $\leq$)$[k]$.*
2. *[22] For all $k$, "Has a $k+1$-Clique" $\notin$ VAR(w.o. $\leq$)$[k]$.*

**Theorem 5.6** *[22] For all $k$,*

$$\textit{Graph Isomorphism} \notin \textit{VAR(w.o. } \leq\textit{)}[k]\,.$$

## 5.3 Replacing Ordering by Counting

Proposition 5.2 shows that we can't prove interesting lower bounds on the computational complexity of a property $S$ simply by constructing a pair of structures which differ on $S$, but are equivalent for a language that includes ordering. On the other hand, the following proposition points out that first-order logic without ordering is not powerful enough to simulate Turing machines, or even to count. Thus, lower bounds on FO(w.o.$\leq$) do not immediately carry over to complexity lower bounds.

**Proposition 5.7** *[27] Let $G$ and $H$ be graphs with $n$ and $n+1$ vertices, respectively, and no edges. Then $G$ and $H$ are $\mathcal{L}_n$ equivalent (in the language without ordering).*

Proposition 5.7 points out the only defect of FO(w.o.$\leq$) that we know about, namely that this language is not strong enough to count. A possible fix is to add counting quantifiers: for each $k \in \mathbf{N}$ add the quantifier $(\exists n\, x)$ where $(\exists n\, x)\varphi(x)$ means that there exists at least $n$ distinct elements $x$ for which $\varphi(x)$ holds. Note that $(\exists n+1\, x)(x = x)$ says that there exist at least $n+1$ points, and only uses one variable instead of the $n+1$ that are needed if we don't have counting quantifiers. Define $C_k$ to be the first-order language $L_k$ plus counting quantifiers. In the rest of this section we will confine our attention to languages without ordering.

In [27] we consider the following algorithms to test whether graphs $G$ and $H$ are isomorphic: For appropriate $k$, test if $G$ is $C_k$ equivalent to $H$.

**Theorem 5.8** *[27] Given graphs $G$ and $H$ with at most $n$ vertices, we can test if $G$ is $\mathcal{L}_k$ equivalent to $H$ and if $G$ is $C_k$ equivalent to $H$ in $DTIME[(n^k \log n)k^2]$ .*

**Definition 5.9** *Let $\Sigma$ be a set of graphs. Define $var(\Sigma, n)$ (resp. $vc(\Sigma, n)$) to be the minimum $k$ such that for all $G, H \in \Sigma$ with $|G|, |H| \leq n$, If $G$ is $\mathcal{L}_k$ (resp. $C_k$) equivalent to $H$, then $G \cong H$. Let $var(n) = var(GRAPHS, n)$ and $vc(n) = vc(GRAPHS, n)$ where $GRAPHS$ is the set of all finite graphs. When $var(\Sigma, n)$ or $vc(\Sigma, n)$ is bounded, we write $var(\Sigma) = \max_n var(\Sigma, n)$, and $vc(\Sigma) = \max_n vc(\Sigma, n)$.*

**Corollary 5.10** *[27] Let $\Sigma$ be any set of graphs, and let $v(n) = vc(\Sigma, n)$. Then we can test isomorphism of graphs in $\Sigma$ in $DTIME[(n^{v(n)} \log n)(v(n))^2]$*

.

**Open Problem 11** *[27] We don't know much about the values of $var(n)$ and $vc(n)$. We know by Theorem 5.6 that $var(n)$ is unbounded. We know that $3 \leq vc(n) \leq n-1$. However, both for its own sake and also in view of Corollary 5.10 it is important to find much better bounds on var and vc.*

We also have very little information right now about the values of var($\Sigma$) and vc($\Sigma$) for most classes of graphs, $\Sigma$. For trees, however, the situation is simpler:

**Theorem 5.11** *Let TREES be the set of all finite trees. Let $T_k$ be the set of finite trees such that each node has at most $k$ children, and let $S_k$ be the subset of $T_k$ in which each non-leaf has exactly $k$ children. Then,*

1. *[26]*

$$var(T_k) = \begin{cases} 2 & \text{if } k = 1 \\ 3 & \text{if } 2 \leq k \leq 3 \\ k & \text{if } k > 3 \end{cases}$$

2. *[26]*

$$var(S_k) = \begin{cases} 2 & \text{if } 1 \leq k \leq 2 \\ 3 & \text{if } 3 \leq k \leq 6 \\ \lceil k/2 \rceil & \text{if } k > 6 \end{cases}$$

3. *[27]*

$$vc(TREES) = 2$$

**Open Problem 12** *[27] It would be very interesting to determine $var(\Sigma, n)$ and $vc(\Sigma, n)$ for various $\Sigma$, for example: planar graphs, trivalent graphs, graphs with bounded color class size, graphs with bounded eigenvalue multiplicity, etc. It would be valuable simply to determine whether or not these values are bounded, cf. Problem 15.*

The problem of appropriately replacing '$\leq$' is a fundamental one. In particular it is not recursively decidable if a given polynomial-time Turing machine, M, computes a graph property or not, i.e. does M accept its inputs irrespective of their ordering? The following question is open.

**Open Problem 13** *Is there an r.e. listing of all polynomial-time graph properties?*

One way to attack Problem 13 is to express problems in languages such as $C_k$; and we would love to know if $C_k$ is rich enough to express all polynomial-time graph properties. One defect of $C_k$ as a language for expressing computation, rather than testing graph isomorphism, is that the arguments to the counting quantifiers are constants. Instead we can consider a two-sorted language with variables $m_i$ ranging over the numbers $0, \ldots, n-1$, with $\leq$ given over this domain (and thus PLUS and TIMES and BIT definable). The connection between the number domain and the universe of the structure would be the counting quantifiers: $(\exists m_i\, x_j)$. Let FOC$[t(n)]$ be the analogue of FO$[t(n)]$ with these numbers and counting quantifiers available. It's not

hard to show that we would still have $\bigcup_i \mathrm{FOC}[n^i] = (FOC + LFP)$. The following three problems now address the question, "Is Counting Enough, or Do We Need Something Else?"

**Open Problem 14** *Prove or disprove: (FOC + LFP) = P.*[4]

Note that a positive solution to Problem 14 would provide a positive solution to Problem 13.

**Open Problem 15** *Prove or disprove: For every set of graphs $\Sigma$ having a polynomial-time graph isomorphism algorithm, $vc(\Sigma, n) = O[1]$.*

**Open Problem 16** *Prove or disprove: Problems 14 and 15 are equivalent.*

# References


[1] M. Ajtai, "$\Sigma_1^1$ Formulae on Finite Structures," *Annals of Pure and Applied Logic* **24**, 1983, (1-48). 603-617.

[2] László Babai and György Turán, "The Complexity of Defining a Relation on a Finite Graph," to appear in *Zeitschr. fur Math. Logik* .

[3] David Barrington, "Bounded-Width Polynomial-Size Branching Programs Recognize Exactly Those Languages in $\mathrm{NC}^1$," *18th ACM STOC* (1986), 1-5.

[4] Jon Barwise, "On Moschovakis Closure Ordinals," J. Symb. Logic 42 (1977), 292-296.

[5] Samuel R. Buss and Neil Immerman, "Iterated Quantifier Blocks and First-Order Logic for Finite Structures," in preparation.

[6] Ashok Chandra and David Harel, "Structure and Complexity of Relational Queries," *21st IEEE Symp. on Foundations of Computer Science,* (1980), (333-347). Also appeared in a revised form in *JCSS* **25** (1982), (99-128).


[4] Steve Lindell [29] has recently shown that if we restrict our attention to complete binary trees, then (FOC + LFP) = P. Lindell also shows that a restriction of (FOC + LFP), in which elements from the number domain may not appear in relations defined by induction, is equal to (FO(w.o.$\leq$) + LFP) and is strictly contained in P, c.f. Theorem 5.11.

[7] Ashok Chandra, Dexter Kozen, and Larry Stockmeyer, "Alternation," *JACM,* **28**, No. 1, (1981), 114-133.

[8] Ashok Chandra, Larry Stockmeyer and Uzi Vishkin, "Constant Depth Reducibility," *SIAM J. of Comp.* **13**, No. 2, 1984, (423-439).

[9] Stephen Cook, "A Taxonomy of Problems with Fast Parallel Algorithms," *Information and Control* **64** (1985), 2-22.

[10] Michel de Rougemont, "Uniform Definability on Finite Structures with Successor," *16th ACM STOC Symp.*, (1984), 409-417.

[11] A. Ehrenfeucht, "An Application of Games to the Completeness Problem for Formalized Theories," Fund. Math. 49 (1961), 129-141.

[12] H. Enderton, *A Mathematical Introduction to Logic,* Academic Press, 1972.

[13] Ron Fagin, "Generalized First-Order Spectra and Polynomial-Time Recognizable Sets," in *Complexity of Computation,* (ed. R. Karp), *SIAM-AMS Proc. 7,* 1974, (27-41).

[14] R. Fraisse, "Sur les Classifications des Systems de Relations," Publ. Sci. Univ. Alger I (1954).

[15] M. Furst, J.B. Saxe, and M. Sipser, "Parity, Circuits, and the Polynomial-Time Hierarchy," *22nd IEEE FOCS Symp.*, 1981, (260-270).

[16] Joachim von zur Gathen, "Permanent and Determinant," it 27th IEEE Symp. on Foundations of Computer Science, (1986), 398-401.

[17] Etienne Grandjean, "The Spectra of First-Order Sentences and Computational Complexity," *SIAM J. of Comp.* **13**, No. 2, 1984, (356-373).

[18] Yuri Gurevich, "Toward Logic Tailored for Computational Complexity," *Computation and Proof Theory* (M.M. Richter et. al., eds.). Springer-Verlag Lecture Notes in Math. 1104 (1984), 175-216.

[19] Yuri Gurevich and Saharon Shelah, "Fixed-Point Extensions of First-Order Logic," *26th Symp. on Foundations of Computer Science,* 1985. (137-161).

[20] Johan Hastad, "Almost Optimal Lower Bounds for Small Depth Circuits," *18th ACM STOC Symp.*, (1986), 6-20.

[21] Neil Immerman, "Number of Quantifiers is Better than Number of Tape Cells," *JCSS* **22**, No. 3, June 1981, 65-72.

[22] Neil Immerman, "Upper and Lower Bounds for First Order Expressibility," *JCSS* **25**, No. 1 (1982), 76-98.

[23] Neil Immerman, "Relational Queries Computable in Polynomial Time," *14th ACM STOC Symp.*, (1982), 147-152. Also appeared in revised form in *Information and Control*, 68 (1986), 86-104.

[24] Neil Immerman, "Languages Which Capture Complexity Classes," *15th ACM STOC Symp.*, (1983) 347-354. Also to appear in revised form in *SIAM J. Comput.* **16**, No. 4 (1987).

[25] Neil Immerman, "Expressibility and Parallel Complexity," Tech. Report, Yale University Department of Computer Science (1987).

[26] Neil Immerman and Dexter Kozen, "Definablitity with Bounded Number of Bound Variables," *Second LICS Symp.* (1987)

[27] Neil Immerman and Eric S. Lander, "Telling Graphs Apart: A First-Order Approach to Graph Isomorphism," Tech. Report, Yale University Department of Computer Science (1987).

[28] Daniel Leivant, "Characterization of Complexity Classes in Higher-Order Logic," *this volume.*

[29] Steven Lindell, personal communication.

[30] L. Lovász and P. Gács, "Some Remarks on Generalized Spectra," *Zeitchr. f. math, Logik und Grundlagen d. Math,* Bd. 23, 1977, (547-554).

[31] James Lynch, "Complexity Classes and Theories of Finite Models," *Math. Sys. Theory* **15**, 1982, (127-144).

[32] Yiannis N. Moschovakis, *Elementary Induction on Abstract Structures,* North Holland, 1974.

[33] Larry Ruzzo, "On Uniform Circuit Complexity," *J. Comp. Sys. Sci.*, **21**, No. 2 (1981), 365-383.

[34] Michael Sipser, "Borel Sets and Circuit Complexity," *15th Symp. on Theory of Computation,* 1983, (61-69).

[35] S. Skyum and L.G. Valiant, "A Complexity Theory Based on Boolean Algebra," *22nd IEEE Symp. on Foundations of Computer Science,* 1981, (244-253). Revised version appears in *JACM,* 32, No. 2, April, 1985, (484-502).

[36] Larry Stockmeyer, "The Polynomial-Time Hierarchy," *Theoretical Comp. Sci.* **3**, 1977,(1-22).

[37] Larry Stockmeyer and Uzi Vishkin, "Simulation of Parallel Random Access Machines by Circuits," *SIAM J. of Comp.* **13**, No. 2, 1984, (409-422).

[38] Martin Tompa, "$A\Sigma_2^L \subseteq A\Pi_2^L$ (Lange, Jenner, and Kirsig)," handwritten notes (Dec. 1986).

[39] György Turán, "On the Definability of Properties of Finite Graphs," *Discrete Math,* **49** (1984), 291-302.

[40] L.G. Valiant, "Reducibility By Algebraic Projections," *L'Enseignement mathématique,* **T. XXVIII**, 3-4, 1982, (253-268).

[41] M. Vardi, "Complexity of Relational Query Languages," *14th Symposium on Theory of Computation,* 1982, (137-146).

[42] Andrew Chi-Chih Yao,"Separating the Polynomial-Time Hierarchy by Oracles," *26th IEEE Symp. on Foundations of Comp. Sci.,* 1985.