SIAM J. COMPUT.
Vol. 16, No. 4, August 1987

011

LANGUAGES THAT CAPTURE COMPLEXITY CLASSES*

NEIL IMMERMAN†

Abstract. We present a series of operators of apparently increasing power which when added to first-order logic produce a series of languages in which exactly the properties checkable in a certain complexity class may be expressed. We thus give alternate characterizations of most important complexity classes. We also introduce reductions appropriate for our setting: first-order translations, and a restricted, quantifier free version of these called projection translations. We show that projection translations are a uniform version of Valiant's projections, and that the usual complete problems remain complete via these very restrictive reductions.

Key words. first-order expressibility, computational complexity, first-order translations, projection translations

AMS(MOS) subject classification. 68025

Introduction. We present in this paper a series of languages adequate for expressing exactly those properties checkable in a series of computational complexity classes. For example, we show that a property of graphs (respectively groups, binary strings, etc.) is in polynomial time if and only if it is expressible in the first order language of graphs (respectively groups, binary strings, etc.) together with a least fixed point operator. As another example, a property is in logspace if and only if it is expressible in first order logic together with a deterministic transitive closure operator.

The roots of our approach to complexity theory go back to 1974 when Fagin showed that the NP properties are exactly those expressible in second order existential sentences. It follows that second order logic expresses exactly those properties which are in the polynomial time hierarchy. We show that adding suitable transitive closure operators to second order logic results in languages capturing polynomial space and exponential time, respectively.

The existence of such natural languages for each important complexity class sheds a new light on complexity theory. These languages reaffirm the importance of the complexity classes as much more than machine dependent issues. Furthermore a whole new approach is suggested. Upper bounds (algorithms) can be produced by expressing the property of interest in one of our languages. Lower bounds may be demonstrated by showing that such expression is impossible.

For example, from the above we know that $P = NP$ if and only if every second order property is already expressible using first order logic plus least fixed point. Similarly nondeterministic logspace is different from P just if there is some sentence using the fixed point operator which cannot be expressed with a single application of transitive closure.

In previous work [Im81], [Im82b], we showed that the complexity of a property is related to the number of variables and quantifiers in a uniform sequence of sentences, $\varphi_1, \varphi_2 \ldots$, where each φ_n expresses the property for structures of size n. Our present formulation is more pleasing because it considers single sentences (in more powerful languages).

* Received by the editors August 29, 1985; accepted for publication (in revised form) September 5, 1986. A preliminary version of this paper appeared in the Proceedings of the 15th Annual ACM Symposium on the Theory of Computing. This research was supported by a National Science Foundation postdoctoral fellowship and by National Science Foundation grant DCR-8603346.

† Computer Science Department, Yale University, New Haven, Connecticut 06520.

The first order expressible properties at first seemed too weak to correspond to any natural complexity class. However we found that a property is expressible by a sequence of first order sentences, $\varphi_1, \varphi_2 \ldots$, where each φ_n has a bounded number of quantifiers if and only if this property is recognized by a similar sequence of polynomial size boolean circuits of bounded depth. It follows that the results of Furst, Saxe, and Sipser, [FSS81], and Sipser, [Si83], translate precisely into a proof that certain properties are not expressible in any first order language.

In this paper we also introduce a reduction between problems that is new to complexity theory. *First order translations*, as the name implies, are fixed first order sentences which translate one kind of structure into another. This is a very natural way to get a reduction, and at the same time it is very restrictive. It seems plausible to prove that such reductions do not exist between certain problems. We present problems which are complete for logspace, nondeterministic logspace, polynomial time, etc., via first order translations.

This paper is organized as follows: Section 1 introduces the complexity classes we will be considering. Section 2 discusses first order logic. Sections 3, 4 and 5 introduce the languages under consideration. Section 6 considers the relationship between first order logic and polynomial size, bounded depth circuits. The present (lack of) knowledge concerning the separation of our various languages is discussed.

1. Complexity classes and complete problems. In this section we define those complexity classess which we will capture with languages in the following sections. We list complete problems for some of the classes. In later sections we will show how to express these complete problems in the appropriate languages; and, we will also show that the problems are complete via first order translations. More information about complexity classes may be found in [AHU74].

Consider the following well-known sequence of containments:

$$L \subseteq \mathrm{NL} \subseteq \Sigma_* L \subseteq P \subseteq \mathrm{NP} \subseteq \Sigma_* P.$$

Here L is deterministic logspace and NL is nondeterministic logspace. $\Sigma_* L = \bigcup_{k=1}^{\infty} \Sigma_k L$ is the logspace hierarchy. Here $\Sigma_k L$ is the class of problems accepted by alternating logspace Turing machines that make $k-1$ alternations between existential and universal states, beginning in an existential state. $\Sigma_* P = \bigcup_{k=1}^{\infty} \Sigma_k P$ is the polynomial time hierarchy, [St77]. Most knowledgeable people suspect that all of the classes in the above containment are distinct, but it is not known that they are not all equal.

We begin our list of complete problems with the graph accessibility problem (GAP). We will consider a *graph* to be a directed graph with vertices $v_0, v_1, v_2, \ldots, v_{\max}$. An *undirected graph* is just a graph whose edge relation is symmetric. Define

$$\mathrm{GAP} = \{G | \exists \text{ a path in } G \text{ from } v_0 \text{ to } v_{\max}\}.$$

THEOREM 1.1. [Sa73]. GAP *is logspace complete for* NL.

We will see later that GAP is complete for NL in a much stronger sense. The GAP problem may be weakened to a deterministic logspace problem by only considering those graphs which have at most one edge leaving any vertex:

$$\mathrm{1GAP} = \{G | G \text{ has outdegree 1 and } \exists \text{ a path in } G \text{ from } v_0 \text{ to } v_{\max}\}.$$

THEOREM 1.2. [HIM78]. 1GAP *is one-way logspace complete for* L.

A problem which lies between 1GAP and GAP in complexity is

$$\text{UGAP} = \{G | G \text{ undirected and } \exists \text{ a path in } G \text{ from } v_0 \text{ to } v_{\max}\}.$$

Let BPL (bounded probability, logspace) be the set of problems, S, such that there exists a logspace coin-flipping machine, M, and if $w \in S$ then Prob(M accepts w) $> 2/3$, while if $w \notin S$ then Prob(M accepts w) $< 1/3$. It follows from the next theorem that UGAP is in BPL. Thus UGAP is probably easier than GAP.

THEOREM 1.3. [AKLLR79]. *If r is a random walk of length $2|E|(|V|+1)$ in an undirected connected graph G then the probability that r includes all vertices in G is greater than or equal to one half.*

Lewis and Papadimitriou [LP82] define *symmetric machines* to be nondeterministic Turing machines whose next move relation on instantaneous descriptions is symmetric. That is if a symmetric machine can move from configuration A to configuration B then it is also allowed to move from B to A. Note that it is not immediately obvious how to build Turing machines that are symmetric in the above sense. Lewis and Papadimitriou show that the definition makes sense. Let Sym-L be the class of problems accepted by symmetric logspace machines. Then we have

THEOREM 1.4. [LP82].

1. L $\subseteq$ Sym-L $\subseteq$ NL.
2. UGAP *is logspace complete for* Sym-L.

John Reif [Re84] extended the notion of symmetric machines to allow alternation. Essentially an alternating symmetric machine has a symmetric next move relation except where it alternates between existential and universal states. Let Σ_*Sym-L $= \bigcup_{k=1}^{\infty} \Sigma_k$Sym-L be the *symmetric logspace hierarchy*. Reif showed that several interesting properties, including planarity for graphs of bounded valence, are in the symmetric logspace hierarchy. It follows that they are also in BPL.

Reif also showed that BPL is contained in $O[\log n]$ time and $n^{O[1]}$ processors on a probabilistic hardware modification machine (HMM).

THEOREM 1.5. [Re84].

$$\Sigma_*\text{Sym-L} \subseteq \text{BPL} \subseteq \text{Prob-HMM-TIME}[\log n],\text{PROC}[n^{O[1]}].$$

Let us prove for example that Σ_2Sym-L is contained in BPL. Given a Σ_2Sym-L machine M and its input, our BPL machine can cycle through all of M's polynomially many universal configurations, a, checking to see that:

1. Configuration a is reachable from the start configuration by a series of exclusively existential moves, and
2. No final, rejecting configuration is reachable from a.

If such a configuration a is found then M accepts, otherwise it rejects. Furthermore, properties (1) and (2) are UGAP questions and therefore answerable by our BPL machine.

One may also consider harder versions of the GAP problem. Let an *alternating graph* $G = (V, E, A)$ be a directed graph whose vertices are labelled universal or existential. $A \subset V$ is the set of universal vertices. Alternating graphs have a different notion of accessibility. Let APATH(x,y) be the smallest relation on vertices of G such that:

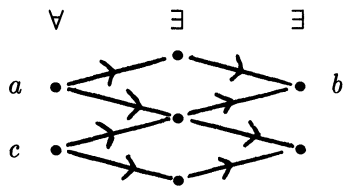

FIG. 1. *An alternating graph.*

1. APATH(x,x).
2. If x is existential and for some edge $\langle x, z\rangle$ APATH(z,y) holds, then APATH(x,y).
3. If x is universal, there is at least one edge leaving x, and for all edges $\langle x, z\rangle$ APATH(z,y) holds, then APATH(x,y).

See Fig. 1 where APATH(a,b) holds, but APATH(c,b) does not. Let

$$\text{AGAP} = \{G | \text{APATH}_G(v_0, v_{\max})\}.$$

It is not hard to see that AGAP is the alternating version of GAP, and thus is complete for ASPACE[log(n)]. Recalling that this class is equal to P [CKS81], we have

THEOREM 1.6. [Im81]. AGAP *is logspace complete for* P.[1]

2. First order logic. In this section we introduce the necessary notions from logic. The reader is referred to [En72] for more background material.

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

For example, if $\tau_0 = \langle \underline{E}^2 \rangle$ consists of a single binary relation symbol then a structure $G = \langle \{0 \ldots n-1\}, E \rangle$ with vocabulary τ_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 $S = \langle \{0 \ldots n-1\}, M \rangle$ with vocabulary τ_1 is a binary string of length n.

If τ is a vocabulary, let

$$\text{STRUCT}(\tau) = \{G | G \text{ is a structure with vocabulary } \tau\}.$$

We will think of a *problem* as a set of structures of some vocabulary τ. 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.

Let the relation $s(x, y)$ be the successor relation on the naturals, i.e., $s(x, y)$ holds iff $x + 1 = y$. Throughout this paper we will assume that $\underline{s}$ is a logical relation symbol denoting the successor relation. We will also assume that *all structures under*

[1] Note that the AGAP problem is easily seen to be equivalent to the monotone circuit value problem shown to be complete for P in [Go77].

consideration have at least two elements. Furthermore the constant symbols $\underline{0}$ and $\underline{\max}$ will always refer to the first and last elements of the universe, respectively. [2]

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 τ and the logical relation symbols and constant symbols: $=, \underline{s}, \underline{0}, \underline{\max}$, using logical connectives: $\wedge, \vee, \neg$, variables: $x, y, z, \ldots$, and quantifiers: $\forall, \exists$.

If $\varphi \in \mathcal{L}(\tau)$ let $\mathrm{MOD}(\varphi)$ be the set of finite models of φ:

$$\mathrm{MOD}(\varphi) = \{G \in \mathrm{STRUCT}(\tau) | G \models \varphi\}.$$

Let FO be the set of all first order expressible problems:

$$\mathrm{FO} = \{S | (\exists \tau)(\exists \varphi \in \mathcal{L}(\tau)) S = \mathrm{MOD}(\varphi)\}.$$

The following result is well known [Fa74], [AU79], [Im81]; but a new proof of the strictness of the containment follows from Corollary 6.3.

THEOREM 2.1. FO *is strictly contained in* L.

3. First order logic plus transitive closure. In view of Theorem 2.1 we wish to strengthen first order logic so that we may express properties of complexity at least logspace . Let $\varphi(x_1 \ldots x_k, x'_1 \ldots x'_k)$ be any formula. It represents a binary relation on k-tuples. We add to our language the operator TC where $\mathrm{TC}[\lambda \bar{x}, \bar{x}' \varphi]$ denotes the reflexive, transitive closure of the relation $\varphi(\bar{x}, \bar{x}')$.[3] Let $(\mathrm{FO} + \mathrm{TC})$ be the set of properties expressible using first order logic plus the operator TC. Let $(\mathrm{FO} + \mathrm{pos\ TC})$ be the set of properties expressible using only positive applications of TC, i.e., not within any negation signs. It is easy to see for example that $\mathrm{GAP} \in (\mathrm{FO} + \mathrm{pos\ TC})$:

$$\mathrm{GAP} = \{G \mid G \models \mathrm{TC}[E(x, x')](0, \max)\}.$$

THEOREM 3.1. $\mathrm{NL} = (\mathrm{FO} + \mathrm{pos\ TC})$.

Proof. It is easy to see that NL contains $(\mathrm{FO} + \mathrm{pos\ TC})$. If $A(x, x')$ can be checked in NL, then so can $\mathrm{TC}[A(x, x')]$: simply guess an A path. Going the other way we are given a nondeterministic logspace Turing machine, M, accepting a set, S, of structures of a certain vocabulary τ. We must produce a sentence ψ in $(\mathrm{FO} + \mathrm{TC})$ such that

$$S = \{G \in \mathrm{STRUCT}(\tau) | G \models \psi\}.$$

The first idea needed to write ψ is that an instantaneous description (ID) of M is of size $O[\log n]$ and can be coded with finitely many variables ranging from 0 to $n - 1$.

For example, suppose that M accepts a graph problem, i.e. the vocabulary, $\tau = \langle \underline{E}^2 \rangle$, consists of a single binary relation symbol. Suppose also that M uses $k \cdot log(n)$ bits of work tape for problems of size n. Input to M consists of n^2 bits – the adjacency matrix for E. An ID for M is a $2k + 3$-tuple: $\langle q, r_1, r_2, w_1, h_1 \ldots w_k, h_k \rangle$. Here q codes M's state and variables r_1, r_2 code the input head position. Note that

[2] As we will see the availability of a successor or ordering relation seems crucial to simulate computation. The last two assumptions are nonstandard and unnecessary. However they are convenient because they make some proofs neater. Otherwise we would have to modify many formulas to deal specially with the trivial case where the universe has only one element. Without the assumption that two distinct constant symbols exist, many irrelevent quantifications would be needed.

[3] Where confusion does not arise we may abbreviate $\mathrm{TC}[\lambda \bar{x}, \bar{x}' \varphi(\bar{x}, \bar{x}')]$ to $\mathrm{TC}[\varphi(\bar{x}, \bar{x}')]$.

the input head is looking at a 1 or 0 according as $E(r_1, r_2)$ holds or does not hold in the input structure. Finally $w_1 ... w_k$ code the $k \cdot log(n)$ bits of M's work tape. One h_i is equal to j where the work head is pointing to the jth bit of w_i; the rest of the h_i's are $n-1$.

The second idea is that using TC we can compute the jth bit of w_i. Let $\mathrm{ON}(w, j)$ mean that $j < \log n$ and bit j of w is on. Starting with the successor relation s we can use TC to express addition and then use TC again to tell if a certain bit in a variable is on:

LEMMA 3.2. *The following predicates are expressible in* (FO + pos TC)*:*

1. $\mathrm{PLUS}(x, y, z) \equiv$ "$x + y = z$".
2. $\mathrm{ON}(w, j)$.

Proof. First, define a binary relation α on pairs such that $\alpha(x, y, x', y')$ holds if and only if $x' = x - 1$ and $y' = y + 1$:

$$\alpha(x, y, x', y') \equiv \big(s(x', x) \wedge s(y, y')\big).$$

Using transitive closure we then get:

$$\mathrm{PLUS}(x, y, z) \equiv \mathrm{TC}[\lambda xy, x'y'\alpha](x, y, 0, z).$$

Now define β as follows:

$$\beta(w, j, w', j') \equiv \big(\exists z(\mathrm{PLUS}(w', w', z) \wedge (w = z \vee s(z, w))) \wedge s(j', j)\big).$$

There is an edge from $\langle w, j\rangle$ to $\langle w', j'\rangle$ if $w' = w/2$ and $j' = j - 1$. Let $\mathrm{ODD}(z)$ abbreviate $\exists x \exists y(\mathrm{PLUS}(x, x, y) \wedge s(y, z))$. It follows that

$$\mathrm{ON}(w, j) \equiv \exists z\big(\mathrm{ODD}(z) \wedge \mathrm{TC}[\lambda wj, w'j'\beta](w, j, z, 0)\big). \qquad \square$$

Once we can tell what the work head is looking at we can write the predicate $\mathrm{NEXT}(\mathrm{ID}_a, \mathrm{ID}_b)$ meaning that ID_b follows from ID_a in one move of M.[4] (Note that the successor relation is used not only to code n bits into a single variable, but also to say that the read head moves one space to the left or right. Any input structure is given to a Turing machine in some order, and it may search the structure in that order.)

Using one more positive application of TC we can express $\mathrm{PATH}(\mathrm{ID}_a, \mathrm{ID}_b)$ meaning that there is a computation of M starting at ID_a and leading to ID_b. Let $\psi = \mathrm{PATH}(\mathrm{ID}_i, \mathrm{ID}_f)$ where ID_i and ID_f are M's initial and final ID's respectively. Then $\psi \in$ (FO+pos TC) and an input structure, G, of the correct vocabulary satisfies ψ if and only if M accepts G. This proves Theorem 3.1. $\square$

The sentence $\psi = \mathrm{PATH}(\mathrm{ID}_i, \mathrm{ID}_f)$ in the above proof has an interesting form. It is written with several positive applications of TC, but we show in the next theorem that these may be merged into one. Thus for each problem C in NL there is a $2k$-ary first order formula φ such that a structure G is a member of C iff G satisfies $\mathrm{TC}[\varphi](\bar{0}, \overline{\mathrm{max}})$. This suggests that GAP is complete for NL via an extremely weak kind of reduction. We call these new reductions *first order translations* and we discuss them right after we show that ψ can indeed be written in this simple form.

THEOREM 3.3. *Let* $\varphi \in$ (FO + pos TC). *Then* φ *is equivalent to a formula of the form*

$$\mathrm{TC}[\psi(u_1, \ldots, u_k, u'_1, \ldots, u'_k)](\bar{0}, \overline{\mathrm{max}})$$

[4] This is a standard construction. See [Fa74], [Im81], [Im82a], [Im82b].

where ψ is first order and $\bar{0}$ (resp. $\overline{\max}$) is the constant symbol 0 (resp. max) repeated k times.

Proof. By induction on the complexity of φ. There are five cases:

1. The formula φ is atomic or the negation of an atomic formula. In this case let u, u' be variables not occurring in φ. Then $\varphi \leftrightarrow \mathrm{TC}[\varphi(u,u')](0,\max)$.
2. $\varphi \equiv \mathrm{TC}[\psi(\bar{u},\bar{u}')](\bar{q},\bar{r})$. We wish to replace $\bar{q},\bar{r}$ with $\bar{0},\overline{\max}$. Put

$$\begin{aligned}\alpha_2(s,t,\bar{u},s',t',\bar{u}') \equiv\ & \big[s=0\wedge t=0\wedge \bar{u}=\bar{0}\ \wedge\ s'=0\wedge t'=\max\wedge\bar{u}'=\bar{q}\big]\\ \vee&\big[s=0\wedge t=\max\wedge\bar{u}\neq\bar{r}\ \wedge\ s'=0\wedge t'=\max\wedge\psi(\bar{u},\bar{u}')\big]\\ \vee&\big[s=0\wedge t=\max\wedge\bar{u}=\bar{r}\ \wedge\ s'\\ &\qquad=\max\wedge t'=\max\wedge\bar{u}'=\overline{\max}\big].\end{aligned}$$

To make this kind of proof more readable we will write formulas such as the above in tabular form. For example, α_2 is described by Table 1.

TABLE 1

α_2	1	2	3
Before s	0	0	0
t	0	max	max
$\bar{u}$	$\bar{0}$	$\neq \bar{r}$	$\bar{r}$
After s'	0	0	max
t'	max	max	max
$\bar{u}'$	$\bar{q}$	s.t. $\psi(\bar{u},\bar{u}')$	$\overline{\max}$

The variables t, s split the ρ-path in three stages: ($st = 00$): Set $\bar{u}'$ to $\bar{q}$ and go to next stage. ($st = 0\max$): Take a ψ step and stay in this stage. When you reach $\bar{r}$ go to next stage. ($st = \max\max$): Set $\bar{u}' = \overline{\max}$ and stop. Thus as desired:

$$\varphi \leftrightarrow \mathrm{TC}[\alpha_2(s,t,\bar{u},s',t',\bar{u}')](\bar{0},\overline{\max})\ .$$

3. $\varphi \equiv \exists x \mathrm{TC}[\lambda\bar{u},\bar{u}'\psi(\bar{u},\bar{u}',x)](\bar{0},\overline{\max})$. Here the notation means that the transitive closure is taken over the relation $\psi(\bar{u},\bar{u}')$ and the variable x occurs free in ψ. Define a formula $\alpha_3(\bar{u},x,\bar{u}',x')$ via Table 2.

TABLE 2

α_3	1	2	3
Before $\bar{u}$	$\bar{0}$	$\neq \overline{\max}$	$\overline{\max}$
x	0	any	any
After $\bar{u}'$	$\bar{0}$	s.t. $\psi(\bar{u},\bar{u}',x)$	$\overline{\max}$
x'	any	x	max

Notice that α_3 allows a guess of x' on the first step. This x must remain until we reach $\overline{\max}$. Thus

$$\varphi \leftrightarrow \mathrm{TC}[\alpha_3(\bar{u},x,\bar{u}',x')](\bar{0},\overline{\max})\ .$$

4. $\varphi \equiv \forall x \mathrm{TC}[\lambda\bar{u},\bar{u}'\psi(\bar{u},\bar{u}';x)](\bar{0},\overline{\max})$. In this case we simulate the $\forall x$ by searching through all x's in order using successor. I'm not sure whether the result holds if s is not available. (See Table 3.)

TABLE 3

α_4	1	2
Before $\bar{u}$	$\neq \overline{\max}$	$\overline{\max}$
x	any	$\neq \max$
After $\bar{u}'$	s.t. $\psi(\bar{u}, \bar{u}', x)$	$\bar{0}$
x'	x	s.t. $s(x, x')$

Note that the form of an α_4 path is as follows: Start with $\bar{u} = \bar{0}$ and $x = 0$ and proceed along column 1 with x fixed until $\bar{u} = \overline{\max}$. Then in column 2 we increment x and repeat until we have checked that for all x, $\mathrm{TC}[\lambda\bar{u}, \bar{u}'\psi(\bar{u}, \bar{u}'; x)](\bar{0}, \overline{\max})$ holds. Thus $\varphi \leftrightarrow \mathrm{TC}[\alpha_4(\bar{u}, x, \bar{u}', x')](\bar{0}, \overline{\max})$

5. $\varphi \equiv \mathrm{TC}\big[\lambda\bar{u}, \bar{v}\mathrm{TC}[\lambda\bar{x}, \bar{x}'\psi(\bar{x}, \bar{x}', \bar{u}, \bar{v})](\bar{0}, \overline{\max})\big](\bar{0}, \overline{\max})$. Here we combine two nested TC's into one on α_5 defined in Table 4.

TABLE 4

α_5	1	2	3	4
Before $\bar{u}$	$\bar{0}$	any	any	any
$\bar{v}$	$\bar{0}$	$\neq \bar{0}$	$\neq \overline{\max}$	$\overline{\max}$
$\bar{x}$	$\bar{0}$	$\neq \overline{\max}$	$\overline{\max}$	$\overline{\max}$
After $\bar{u}'$	$\bar{0}$	$\bar{u}$	$\bar{v}$	$\overline{\max}$
$\bar{v}'$	any	$\bar{v}$	any	$\overline{\max}$
$\bar{x}'$	$\bar{0}$	s.t. $\psi(\bar{x}, \bar{x}', \bar{u}, \bar{v})$	$\bar{0}$	$\overline{\max}$

We claim that $\varphi \leftrightarrow \mathrm{TC}[\alpha_5(\bar{u}, \bar{v}, \bar{x}, \bar{u}', \bar{v}', \bar{x}')](\bar{0}, \overline{\max})$. This holds because an α_5 path consists exactly of a series of $\psi(\cdot, \cdot, \bar{u}, \bar{v})$ paths from $\bar{0}$ to $\overline{\max}$ with $\bar{u}, \bar{v}$ fixed (column 2). At the end of any such path we know that $\mathrm{TC}[\lambda\bar{x}, \bar{x}'\psi(x, x'; u, v)](\bar{0}, \overline{\max})$ holds. The α_5 path may now appropriately step from $(\bar{u}, \bar{v}, \overline{\max})$ to $(\bar{v}, \bar{v}', \bar{0})$, i.e. reach $\bar{v}$ and begin trying to move from $\bar{v}$ to some new $\bar{v}'$ (column 3). Finally if we ever reach $\bar{x} = \overline{\max}$ when $\bar{v} = \overline{\max}$ then we are done (column 4).

Note that the cases of disjunction and conjunction follow easily from cases 3 and 4, respectively. □

DEFINITION 3.4. Let τ_1 and τ_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*, σ, of $\mathcal{L}(\tau_2)$ in $\mathcal{L}(\tau_1)$ is a sequence of r formulas from $\mathcal{L}(\tau_1)$: $\Sigma_i(x_{11} \ldots x_{1k} \ldots x_{a_i k})$ $i = 1 \ldots r$, where each Σ_i has $k \cdot a_i$ free variables.

Such an interpretation σ translates any structure $G \in \mathrm{STRUCT}[\tau_1]$ to a structure $\sigma(G) \in \mathrm{STRUCT}[\tau_2]$ where the universe of $\sigma(G)$ is the set of all k-tuples from the universe of G, i.e., $|\sigma(G)| = |G|^k$. The successor relation on $\sigma(G)$ is defined using the lexicographical ordering on these k-tuples. Thus $0^{\sigma(G)} = \langle 0^G, \ldots, 0^G \rangle$ and $\max^{\sigma(G)} = \langle \max^G, \ldots, \max^G \rangle$. Each nonlogical relation R_i of $\sigma(G)$ is defined using the corresponding Σ_i:

$$R_i^{\sigma(G)} = \{\langle \bar{g}_1, \ldots, \bar{g}_{a_i} \rangle \in |\sigma(G)|^{a_i} \mid G \models \Sigma_i(\bar{g}_1, \ldots, \bar{g}_{a_i})\}.$$

See [En72] for a discussion of interpretations between theories. See also [LG77] for a proof that SAT is NP complete via "elementary reductions" – essentially the same thing as our first order translations.

DEFINITION 3.5. Given two problems: $S \subset \mathrm{STRUCT}[\tau_1]$ and $T \subset \mathrm{STRUCT}[\tau_2]$, a *first order translation* of S to T is an interpretation, σ, of $\mathcal{L}(\tau_2)$ in $\mathcal{L}(\tau_1)$ such that:

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

As our first example we prove:

COROLLARY 3.6. GAP *is complete for* NL *via first order translations.*

Proof. Let $S \subseteq \text{STRUCT}[\tau_1]$ be any problem in NL. By Theorems 3.1 and 3.3 there is a constant k and a formula $\psi(u_1, \ldots, u_k, u'_1, \ldots, u'_k) \in \mathcal{L}(\tau_1)$ such that given $\mathcal{A} \in \text{STRUCT}[\tau_1]$ we have

$$\mathcal{A} \in S \Leftrightarrow \mathcal{A} \models \text{TC}[\psi](\bar{0}, \overline{\max}) \quad .$$

Let $\tau_2 = \langle \underline{E}^2 \rangle$. Let $\sigma = \langle \psi(\bar{u}, \bar{u}') \rangle$ be an interpretation of $\mathcal{L}(\tau_2)$ in $\mathcal{L}(\tau_1)$. Then σ is a first order translation of S to GAP because for any $\mathcal{A} \in \text{STRUCT}[\tau_1]$ we have:

$$\begin{aligned} \mathcal{A} \in S \quad &\Leftrightarrow \quad \mathcal{A} \models \text{TC}[\lambda \bar{u}, \bar{u}' \psi(\bar{u}, \bar{u}')](\bar{0}, \overline{\max}) \\ &\Leftrightarrow \quad \sigma(\mathcal{A}) \models \text{TC}[\lambda x, x' E(x, x')](0, \max) \\ &\Leftrightarrow \quad \sigma(A) \in \text{GAP} \, . \end{aligned}$$

□

The first order translation is an extremely weak kind of reduction and it is clearly appropriate for comparing the power of logical languages. It follows from Theorem 6.2 that first order translations are a uniform version of constant depth reductions, cf. [CSV84]. It would seem that projection reductions [SV85] are even weaker. However an examination of the proof of Theorem 3.3 reveals that not only have we removed all but one occurrence of TC, but we have removed all the quantifiers as well. In fact we have proved the following stronger version of Theorem 3.3.

THEOREM 3.7. *Let* $\varphi \in (\text{FO} + \text{pos TC})$. *Then* φ *is equivalent to a formula of the form* $\text{TC}[\psi](\bar{0}, \overline{\max})$ *where* ψ *is a quantifier free formula in disjunctive normal form:*

$$\psi \equiv \bigvee_{i=1}^{I} \alpha_i \wedge \beta_i$$

where

1. *Each* α_i *is a conjunction of the logical atomic relations,* $s, =$, *and their negations.*
2. *Each* β_i *is atomic or negated atomic.*
3. *For* $i \neq j$, α_i *and* α_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 ψ in the above theorem. Note that if ψ is represented in tabular form then the above conditions are equivalent to the following:

1. No quantifiers and no logical connectives except '$\neg$' occur in the table.
2. There is at most one occurrence of a nonlogical relation symbol in each column.
3. The columns are mutually exclusive.

It follows immediately that

COROLLARY 3.8. GAP *is complete for* NL *via projection translations.*

We hope that those readers familiar with the projection reductions of [SV85] will check for themselves that projection translations are a uniform version of projection reductions.

We close this section with a discussion of the complexity class (FO + TC). We mistakenly claimed in [Im83] that the equality $\Sigma_* L = (\text{FO} + \text{TC})$ could be derived

simply by closing both sides of the equation of Theorem 3.1 under negation. Unfortunately this is wrong, and we now suspect that the two classes are distinct. We can prove the following:

THEOREM 3.9.

$$\bigcup_{k=1}^{\infty} \Sigma_k L \subseteq (\mathrm{FO} + \mathrm{TC}).$$

Proof. Let M be a $\Sigma_k L$ Turing machine and let $\mathcal{A}$ be an input structure to M. Let the predicates $\mathrm{EPATH}_M(\bar{x}, \bar{y})$ (resp. $\mathrm{APATH}_M(\bar{x}, \bar{y})$) mean that $\bar{x}$ and $\bar{y}$ are ID's of M such that there is a computation path of M each of whose ID's is existential (resp. universal) except for $\bar{y}$ which is universal or final (resp. existential or final). It is immediate from Theorem 3.1 that EPATH and APATH are expressible in (FO + pos TC). Let ID_i and ID_f be M's initial and final ID's. Then M accepts $\mathcal{A}$ if and only if $\mathcal{A}$ satisfies the following sentence:

$$(\exists \mathrm{ID}_1 \forall \mathrm{ID}_2 \exists \mathrm{ID}_3 \ldots \mathrm{Q}_k \mathrm{ID}_k) \quad \Big[\mathrm{EPATH}(\mathrm{ID}_i, \mathrm{ID}_1) \wedge \big[\neg \mathrm{APATH}(\mathrm{ID}_1, \mathrm{ID}_2) \vee [\mathrm{EPATH}(\mathrm{ID}_2, \mathrm{ID}_3) \wedge \ldots \wedge \mathrm{ID}_k = \mathrm{ID}_f] \ldots\big]\Big]. \qquad \square$$

Note that the above proof requires no nesting of TC's and $\neg$TC's which is why we suspect that $\Sigma_* L \neq (\mathrm{FO} + \mathrm{TC})$.

4. Other transitive closure operators. We next employ other operators not quite as strong as TC to capture weaker complexity classes. For example, let STC be the symmetric transitive closure operator. Thus if $\varphi(\bar{x}, \bar{x}')$ is a first order relation on k-tuples, then $\mathrm{STC}[\lambda \bar{x}, \bar{x}' \varphi]$ is the symmetric, transitive closure of φ.

$$\mathrm{STC}[\lambda \bar{x}, \bar{x}' \varphi] \equiv \mathrm{TC}[\lambda \bar{x}, \bar{x}' \varphi(\bar{x}, \bar{x}') \vee \varphi(\bar{x}', \bar{x})].$$

The following theorem, whose proof is similar to the proof of Theorem 3.1, shows that STC captures the power of symmetric logspace:

THEOREM 4.1.

1. Sym-L = (FO + pos STC).
2. UGAP *is complete via first order translations for* Sym-L.

We can also add a deterministic version of transitive closure which we call DTC. Given a first order relation $\varphi(\bar{x}, \bar{x}')$ let

$$\varphi_d(\bar{x}, \bar{x}') \equiv \varphi(\bar{x}, \bar{x}') \wedge [(\forall \bar{z}) \neg \varphi(\bar{x}, \bar{z}) \vee (\bar{x}' = \bar{z})].$$

That is, $\varphi_d(\bar{x}, \bar{x}')$ is true just if $\bar{x}'$ is the unique descendent of $\bar{x}$. Now define:

$$\mathrm{DTC}(\varphi) \equiv \mathrm{TC}(\varphi_d).$$

Note that (FO + pos DTC) is closed under negation. Thus:

THEOREM 4.2.

1. $L = (\mathrm{FO} + \mathrm{DTC})$.
2. 1GAP *is complete for* L *via first order translations.*

To prove Theorem 4.1 (resp. Theorem 4.2) we must check that the proofs of Theorem 3.1, Lemma 3.2, and Theorem 3.3 go through when we replace TC by STC

(resp. DTC). Notice that the occurrences of TC in the proof of Lemma 3.2 are of the form $\mathrm{TC}[\gamma](\bar{s},\bar{t})$ where for every tuple $\bar{x}$ there is at most one $\bar{x}'$ such that $\gamma(\bar{x},\bar{x}')$ and furthermore there is no $\bar{x}'$ such that $\gamma(\bar{t},\bar{x}')$. It follows that

$$\mathrm{TC}[\gamma](\bar{s},\bar{t}) \equiv \mathrm{DTC}[\gamma](\bar{s},\bar{t}) \equiv \mathrm{STC}[\gamma](\bar{s},\bar{t}) \quad .$$

Thus Lemma 3.2 still holds with STC or DTC replacing TC as needed. See [LP82] for a more detailed discussion of when nondeterministic (TC) computations or deterministic (DTC) computations may be replaced by symmetric (STC) computations.

To finish the proof of part (1) of Theorems 4.1 and 4.2 we follow the proof of Theorem 3.1. We can express $\mathrm{NEXT}(\mathrm{ID}_a,\mathrm{ID}_b)$ using ON and no additional uses of TC. Finally to express PATH from NEXT one additional use of the STC for Theorem 4.1 or DTC for Theorem 4.2 suffices.

To prove part (2) of Theorems 4.1 and 4.2 we observe that Theorem 3.3 holds when TC is replace by STC or DTC. First consider STC. The reader should check that the theorem and proof remain true if we replace TC everywhere by STC. A typical example is case 4. Here $\varphi \equiv (\forall x)\mathrm{STC}[\lambda\bar{u},\bar{u}'\psi(\bar{u},\bar{u}',x)](\bar{0},\overline{\max})$. Recall that we defined α_4 by Table 5.

TABLE 5

α_4	1	2
Before $\bar{u}$	$\neq \overline{\max}$	$\overline{\max}$
x	any	$\neq \max$
After $\bar{u}'$	s.t. $\psi(\bar{u},\bar{u}',x)$	$\bar{0}$
x'	x	s.t. $s(x,x')$

We claim that:

$$\varphi \leftrightarrow \mathrm{STC}[\alpha_4(\bar{u},x,\bar{u}',x')](\bar{0}0,\overline{\max}\max) \; .$$

To prove this last equivalence suppose that φ holds. Then for all x there is a symmetric $\psi(\cdot,\cdot,x)$ path p_x from $\bar{0}$ to $\overline{\max}$. It follows that

$$(p_0 \times 0),\ \langle(\overline{\max},0),(\bar{0},1)\rangle,\ (p_1 \times 1),\ \ldots,\ \langle(\overline{\max},\max-1),(\bar{0},\max)\rangle,\ (p_{\max} \times \max)$$

is an α_4 path from $\bar{0}0$ to $\overline{\max}\max$. Conversely, any minimal α_4 path from $\bar{0}0$ to $\overline{\max}\max$ must include the edges,

$$\langle(\overline{\max},0),(\bar{0},1)\rangle,\ \langle(\overline{\max},1),(\bar{0},2)\rangle,\ldots,\langle(\overline{\max},\max-1),(\bar{0},\max)\rangle$$

in that order. Furthermore the part of the path between any consecutive pair of the above edges will have constant second component. But this just says that for all x, $\mathrm{STC}[\psi(\bar{u},\bar{u}',x)[\bar{0}0,\overline{\max}\max]]$, i.e., φ holds.

In the DTC case we must modify the construction of the proof of Theorem 3.3 so that a deterministic path is not transformed into a nondeterministic path. The most interesting case is the existential quantifier:

$$\varphi \equiv \exists x\mathrm{DTC}[\lambda\bar{u},\bar{u}'\psi(\bar{u},\bar{u}',x)](\bar{0},\overline{\max}).$$

Here instead of letting the path finder guess the correct x we force the path to try all x's and go to $\overline{\max}$ when a correct one is found. We use the fact that there is a path in an n^k vertex graph if and only if there is such a path of length at most n^k-1.

Let arity($\bar{z}$) = arity($\bar{w}$) = arity($\bar{u}$). In the following z is a counter used to cut off a cycling α-path and w is used to find the α-edge leaving u if one exists. We will abuse notation and write '$s(\bar{z}, \bar{z}')$' to mean that $\bar{z}'$ is the successor of $\bar{z}$ in the lexicographical ordering induced by the successor relation s. Define α_3' by Table 6.

TABLE 6

α_3'	1	2	3	4	5
Before	s.t. $\psi(\bar{u}, \bar{w}, x)$	s.t. $\neg\psi(\bar{u}, \bar{w}, x)$	s.t. $\neg\psi(\bar{u}, \bar{w}, x)$		
$\bar{u}$	$\neq \overline{\max}$	$\neq \overline{\max}$	$\neq \overline{\max}$	$\neq \overline{\max}$	$\overline{\max}$
$\bar{z}$	$\neq \overline{\max}$	$\neq \overline{\max}$	$\neq \overline{\max}$	$\overline{\max}$	any
$\bar{w}$	any	$\neq \overline{\max}$	max	any	any
x	any	any	$\neq$ max	$\neq$ max	any
After $\bar{u}'$	$\bar{w}$	$\bar{u}$	$\bar{0}$	$\bar{0}$	$\overline{\max}$
$\bar{z}'$	s.t. $s(\bar{z}, \bar{z}')$	$\bar{z}$	$\bar{0}$	$\bar{0}$	$\overline{\max}$
$\bar{w}'$	$\bar{0}$	s.t. $s(\bar{w}, \bar{w}')$	$\bar{0}$	$\bar{0}$	$\overline{\max}$
x'	x	x	s.t. $s(x, x')$	s.t. $s(x, x')$	$\overline{\max}$
Comment	take edge to $\bar{w}$; increment $\bar{z}$	no edge from $\bar{u}$ to $\bar{w}$: try next $\bar{w}$	no exit from $\bar{u}$: try next x	looping: try next x	The path is found!

Intuitively, an α_3' path from $\langle \bar{u}, \bar{z}, \bar{w}, x\rangle$ moves to $\langle \bar{w}, \bar{z}+1, \bar{0}, x\rangle$ if $\psi(\bar{u}, \bar{w}, x)$. Otherwise we increment $\bar{w}$ until such an ψ-edge is found. If none is found (column 3), or if we are in a loop (column 4) then we try the next x until we finally reach $\overline{\max}$ or exhaust all x's. It follows that:

$$\varphi \leftrightarrow \mathrm{DTC}[\alpha_3'(\bar{u}, \bar{z}, \bar{w}, x, \bar{u}', \bar{z}', \bar{w}', x')](\bar{0}, \overline{\max}) .$$

This completes the proofs of Theorems 4.1 and 4.2. □

In spite of an over optimistic claim in [Im83] we are not sure whether Σ_*Sym-L = (FO + STC) but we suspect that equality does *not* hold. We can prove the following:

THEOREM 4.3.

$$\Sigma_*\text{Sym-L} \subseteq (\text{FO} + \text{STC}) \subseteq \text{BPL}$$

Proof. The first inclusion follows as in the proof of Theorem 3.9. The second inclusion follows from Theorem 1.3 and standard techniques, (see for example [Re84]). Once we know that the predicate $\alpha(\bar{x}, \bar{x}')$ is in BPL, we can test if $\alpha(\bar{x}, \bar{x}')$ holds with exponentially low probablity of error. Then we can take a random α walk and use Theorem 1.3 to see that STC$[\alpha(\bar{x}, \bar{x}')]$ is also testable in BPL. □

The penultimate operator we add in this section is least fixed point, LFP. Given a first order operator on relations:

$$\varphi(R)[\overline{x}] \equiv Q_1 z_1 \ldots Q_k z_k M(\overline{x}, \overline{z}, R)$$

we say that φ is *monotone* if $R_1 \subset R_2$ implies $\varphi(R_1) \subset \varphi(R_2)$. For a monotone φ define:

$$\mathrm{LFP}(\lambda \bar{x}\varphi) \equiv \min\{R | \varphi(R) = R\}$$

It is well known that LFP(φ) exists and is computable in polynomial time in the size of the structure involved, see e.g. [Im82b].

EXAMPLE 4.4. The least fixed point operator is a way of formalizing inductive definitions of new relations. Recall the AGAP property discussed in Section 1. Consider the following monotone operator:

$$\Gamma(R)[x,y] \equiv \quad (x=y) \vee \Big[(\exists z)(E(x,z) \wedge R(z,y)) \\ \wedge\big(\neg A(x) \vee (\forall z)(\neg E(x,z) \vee R(z,y))\big)\Big].$$

It is easy to see that

$$\mathrm{LFP}(\Gamma) = \mathrm{APATH}$$

and in fact:

THEOREM 4.5. [Im82a],[Va82]. $P = (\mathrm{FO} + \mathrm{LFP})$.

Instead of using LFP we introduce a variant of it that is more in keeping with DTC, STC, and TC. Suppose that $\varphi(\bar{x},\bar{x}')$ and $\alpha(\bar{x})$ are given formulas where $\mathrm{arity}(x) = \mathrm{arity}(y) = k$. For a given structure $\mathcal{A}$ these formulas define an alternating graph $G_{\varphi,\alpha}$ whose universe consists of k-tuples from $\mathcal{A}$, whose edge relation is the set of pairs of k-tuples for which φ holds in $\mathcal{A}$, and whose universal nodes are those k-tuples satisfying α. We define the *alternating transitive closure operator* (ATC) to be the operator whose value on the pair $\varphi(\bar{x},\bar{x}'), \alpha(\bar{x})$ is $\mathrm{APATH}_{G_{\varphi,\alpha}}$. We can define ATC more precisely in terms of LFP as follows. Referring to Example 4.4, let

$$\Gamma(\varphi,\alpha,R)[\bar{x},\bar{x}'] \equiv \quad (\bar{x}=\bar{x}') \vee \Big[(\exists \bar{z})(\varphi(\bar{x},\bar{z}) \wedge R(\bar{z},\bar{x}')) \\ \wedge\big(\neg\alpha(\bar{x}) \vee (\forall \bar{z})(\neg\varphi(\bar{x},\bar{z}) \vee R(\bar{z},\bar{x}'))\big)\Big].$$

DEFINITION 4.6. $\mathrm{ATC}[\lambda\bar{x},\bar{x}'\varphi,\alpha] \equiv \mathrm{LFP}(\bar{x},\bar{x}'\Gamma(\varphi,\alpha))$.

We conclude this section with

THEOREM 4.7.

1. $P = (\mathrm{FO} + \mathrm{ATC})$.
2. AGAP *is complete for P via first order translations.*

Proof. This follows as in the proofs of Theorems 3.1, 4.1 and 4.2, recalling that $\mathrm{ASPACE}[\log n] = P$, [CKS81]. The construction of $\mathrm{NEXT}(\mathrm{ID}_a, \mathrm{ID}_b)$ in the proof of Theorem 3.1 works without change, noting that ATC is at least as powerful as TC. Writing the formula $\alpha(x)$, meaning that ID x is in a universal state, is trivial. The $\mathrm{ASPACE}[\log n]$ machine M accepts a structure $\mathcal{A}$ iff $\mathcal{A} \models \mathrm{ATC}(\mathrm{NEXT},\alpha)(\mathrm{ID}_i, \mathrm{ID}_f)$.

To prove part 2 we just note that Theorem 3.3 remains true with ATC substituted for TC. □

5. Second order logic. In second order logic we have first order logic plus the ability to quantify over relations on the universe. The following theorem of Fagin was our original motivation for this line of research:

THEOREM 5.1. [Fa74]. NP = *(2nd O existential).*

Fagin's theorem says that a property is recognizable in NP iff it is expressible by a second order existential formula. Note that we no longer need "s" as a logical symbol because in second order logic we can say, "There exists a binary relation which is a total ordering on the universe." Closing both sides of Theorem 5.1 under negation gives us that a problem is in the polynomial time hierarchy iff it is expressible in second order logic.

THEOREM 5.2. [St77]. $\Sigma_* P = (2nd\ O)$.

Fagin's original result used $2k$-ary relations to encode the $O[n^{2k}]$ bits of an entire NTIME$[n^k]$ computation. Thus he showed:

$$\text{NTIME}[n^k] \subset (2nd\ O \text{ existential, arity } 2k) \subseteq \text{NP}.$$

Lynch [Ly82] points out that in the presence of addition as a new logical relation on the universe, the second order existential sentences can guess merely the n^k moves and piece together the whole computation in arity k. Thus, he shows:

THEOREM 5.3. [Ly82]. NTIME$[n^k] \subseteq (2nd\ O\ existential\ with$ PLUS, $arity\ k)$.

COROLLARY 5.4.

$$\bigcup_{c=1}^{\infty} \Sigma_c\text{TIME}[n^k] = (2nd\ O\ with\ \text{PLUS}, arity\ k).$$

Proof. Let $S \in \Sigma_c\text{TIME}[n^k]$. It follows that

$$S = \{\mathcal{A} \mid \exists w_1 \forall w_2 \dots Q_{c-1} w_{c-1} T(\mathcal{A}, \bar{w})\}$$

where $w_1, \dots, w_{c-1}$ are bounded in length by $|\mathcal{A}|^k$ and $T(\mathcal{A}, \bar{w})$ is an NTIME$(|\mathcal{A}|^k)$ property if c is odd and a CONTIME$(|\mathcal{A}|^k)$ property if c is even. It follows from Theorem 5.3 that T is expressible as a second order existential, arity k formula: $\psi \equiv \exists R_c \varphi(R_c)$ if c is odd, (or if c is even, $\psi \equiv \forall R_c \varphi(R_c)$). Finally we can replace each w_i of length at most n^k by a relation R_i on $\mathcal{A}$ of arity k. Thus

$$S = \{\mathcal{A} \mid \mathcal{A} \models \exists R_1 \forall R_2 \dots Q_c R_c \psi\}.$$

Conversely, let φ be a second order arity k formula,

$$\varphi \equiv \exists R_1 \forall R_2 \dots Q_c R_c \psi$$

where ψ is first order.

In particular $\psi \equiv (\exists x_1 \dots Q_a x_a) M(\bar{R}, \bar{x})$ where M is a constant size quantifier free formula. It follows that we can test in $\Sigma_{c+a}\text{TIME}[n^k]$ whether an input $\mathcal{A}$ satisfies ψ. This is done by guessing the R's and x's in the appropriate existential or universal state. It remains to check the truth of a constant size quantifier free sentence. This can easily be done in time linear in $|\langle \mathcal{A}, R_1, \dots, R_c \rangle|$. □

Note that in the above results the relation "PLUS" need only be added when k is 1 or 2, otherwise it is definable.

As in the previous section we can add closure operators to second order logic in order to express properties which seem computationally more difficult than the polynomial time hierarchy. If $\varphi(\overline{R}, \overline{S})$ is a sentence expressing a binary super relation on k-tuples of relations $\overline{R}$ and $\overline{S}$, then TC(φ), STC(φ), DTC(φ) express the transitive closure, symmetric transitive closure, deterministic transitive closure, respectively, of φ. It is not hard to show:

THEOREM 5.5. *For* $k = 1, 2, \dots$

1. DSPACE$[n^k] = (2nd\ O$ arity $k, +\text{DTC})$
2. Sym-SPACE$[n^k] = (2nd\ O$ arity $k, +\text{pos STC})$.
3. NSPACE$[n^k] = (2nd\ O$ arity $k, +\text{pos TC})$.

Proof. A k-ary relation R over an n element universe consists of n^k bits. It is an easy exercise to code an $O[n^k]$ space instantaneous description with a set of k-ary relations, $X_1 \dots X_c$, and to write the first order sentence $\varphi_M(X_1 \dots X_c, Y_1 \dots Y_c)$

meaning that ID $\overline{Y}$ follows from $\overline{X}$ in one move of Turing machine M. One aplication of the appropriate transitive closure operator expresses an entire computation. □

The following follows immediately:

COROLLARY 5.6. PSPACE = (2*nd* O + TC) = (2*nd* O + STC) = (2*nd* O + DTC).

In similar fashion we can add a second order ATC operator. Recalling that ASPACE$[n^k] = \bigcup_{c=1}^{\infty}$ DTIME$[c^{n^k}]$ we discover

THEOREM 5.7. *For* $k = 1, 2, \ldots$

$$\bigcup_{c=1}^{\infty} \text{DTIME}[c^{n^k}] = (2nd\ O \text{ arity } k + \text{ATC}).$$

6. Lower bounds. A lower bound by Furst, Saxe and Sipser has interesting consequences for us. (See [FSS81] and also [Aj83] and [Ya85].) The PARITY problem is to determine whether the n inputs to a boolean circuit include an even number which are one.

THEOREM 6.1. [FSS81]. PARITY *cannot be computed by a sequence of polynomial size, bounded depth circuits.*

Theorem 5.1 interests us greatly because of the following relation between bounded depth polynomial size circuits and first order sentences:

THEOREM 6.2. *Given a problem* S *and an integer* $d > 1$ *the following are equivalent:*

1. S *is recognized by a sequence of depth* $d + 1$*, polynomial size circuits, with bounded fan-in at the bottom level.*

2. *There exists a new logical relation* $R \subset \mathbf{N}^a$ *and a first order formula* φ *in which* R *occurs such that* φ *expresses* S*. The formula* φ *contains* d *alternating blocks of quantifiers.*

Proof. (1 $\Rightarrow$ 2): We may assume that the similarity type of S is $\tau = \langle M^1 \rangle$ consisting of a single monadic predicate. Thus to a first order formula an input of size n is a structure $\mathcal{A} = \langle \{0 \ldots n-1\}, M^{\mathcal{A}} \rangle$. To a boolean circuit the same input is the string of n boolean variables $a_1 \ldots a_n$, where $(a_i = 1) \Leftrightarrow i \in M^{\mathcal{A}}$.

Suppose that (1) holds and that the circuits $C_1, C_2, \ldots$ accepting S have depth $d + 1$ and size at most n^k and that their bottom level has fan-in at most k. For definiteness assume that the C_i's top gate is an 'and' gate and that their bottom gate is an 'or' gate. The other cases are analagous. We may also assume – by repeating portions of the circuit if necessary – that the fan-in at all levels of C_n besides the bottom is n^k. We may label each of the n^k edges leaving a gate by a k digit integer in n-ary, $z_1 z_2 \ldots z_k$. In this way each 'or' gate v at the bottom level has a label $z_{11} z_{12} \ldots z_{1k} z_{21} \ldots z_{dk}$. Each such gate in C_n is a disjunction $s_1 \vee \ldots \vee s_k$ of literals: a_i or $\bar{a}_i$.

We wish to define the logical relation R so that for each n, it codes the circuit C_n by listing the literals, $s_1, \ldots, s_k$, of each bottom gate. We will define R in two steps. First we define a relation Q which does the coding. Then we will refine this to R in such a way that we can write the desired formula using only d alternations.

Define the relation Q as follows: $Q(i, z_{11}, z_{12}, \ldots, z_{dk}, y, j)$ holds iff $j = 0$ (resp. $j = \max$) and literal a_y (resp. $\bar{a}_y$) occurs in the gate with label $z_{11} z_{12} \ldots z_{dk}$ in C_{i+1}.

Define the sentence

$$\psi \equiv \forall z_{11}\dots z_{1k}\exists z_{21}\dots z_{2k}\dots\forall z_{d1}\dots z_{dk}\exists y\big(Q(\max,\bar{z},y,0)\wedge M(y)\\ \vee Q(\max,\bar{z},y,\max)\wedge\neg M(y)\big).$$

For inputs of size n, the sentence ψ mimics the circuit C_n. Inside the square brackets it says that one of the disjuncts of the node $z_{11}\dots z_{dk}$ is true. It follows that for any structure $\mathcal{A}$, $\mathcal{A}\models\psi$ if and only if $\mathcal{A}\in S$.

What remains to be shown is that we can remove the quantifier $\exists y$ so that our formula has d alternating blocks of quantifiers as required. We do this by saying that for all $y_1\dots y_k$, if the y_i's are distinct and correspond to all the disjuncts of the 'or' gate in question, then at least one of them is true. We define R as a modification of Q such that for each n and each node $\bar{z}$ there are exactly k y's such that $R(\max,\bar{z},y,0)\vee R(\max,\bar{z},y,\max)$ holds. This is done by throwing in c dummy nodes w for gates with only $k-c$ disjuncts. For each such w we make $R(\max,\bar{z},w,0)$ and $R(\max,\bar{z},w,\max)$ true. That is we define R as follows: For all $n,\bar{z}$ if only $y_1,\dots,y_{k-c}$ satisfy $Q(\max,\bar{z},y,0)\vee Q(\max,\bar{z},y,\max)$, then choose $w_1,\dots,w_c$ distinct from these y_i's (note that we may assume that $k<n$). Let

$$R(\max,\bar{z},\cdot,\cdot)=Q(\max,\bar{z},\cdot,\cdot)\cup\big\{\langle\max,\bar{z},w_i,j\rangle\ \big|\ i=1\dots c;j=0,\max\big\}$$

and put

$$\alpha\equiv\forall y_1\dots\forall y_k\Big(\big[y_1\neq y_2\wedge y_1\neq y_3\wedge\dots\wedge y_{k-1}\neq y_k\\ \wedge(R(\max,\bar{z},y_1,0)\vee R(\max,\bar{z},y_1,\max))\\ \wedge\dots\wedge(R(\max,\bar{z},y_k,0)\vee R(\max,\bar{z},y_k,\max))\big]\\ \rightarrow\big[(\neg R(\max,\bar{z},y_1,0)\wedge\neg M(y_1))\\ \vee(\neg R(\max,\bar{z},y_1,\max)\wedge M(y_1))\vee\dots\vee(\neg R(\max,\bar{z},y_k,\max)\wedge M(y_k))\big]\Big).$$

Here α says that $y_1\dots y_k$ are all the disjuncts of the "or" gate in question and at least one of them comes out true. Finally let

$$\varphi\equiv\forall z_{11}\dots z_{1k}\exists z_{21}\dots z_{2k}\dots\forall z_{d1}\dots z_{dk}(\alpha).$$

(Then φ is as required.)

$(2\Rightarrow 1)$: The converse is simpler. Assume (2). Then the sentence φ must be in the following form:

$$\varphi\equiv\forall z_{11}\dots z_{1k}\exists z_{21}\dots z_{2k}\dots Q_d z_{d1}\dots z_{dk}(\gamma(R,M,\bar{z}))$$

where γ is quantifier free. By replacing each universal (resp. existential) block of quantifiers by an "and" (resp. "or") gate, we can rewrite φ as a circuit of depth d and fan-in n^k in which each bottom gate whose label is the $d\cdot k$ tuple $\bar{c}$ is assigned the value $\gamma(R,M,\bar{c})$. The formula $\gamma(R,M,\bar{c})$ for a fixed assignment of $\bar{c}$ is a boolean combination of formulas of the form $M(c_{ij})$, i.e., a constant size boolean combination of literals independent of n. Each such formula can be written in disjunctive normal form if Q_d is $\exists$ and conjunctive normal form if it's $\forall$. The resulting circuit is polynomial size and depth $d+1$ with the bottom level of bounded fan-in as required. □

Note that the role of the new logical relation R in the above proof is to translate a nonuniform sequence of circuits into a single first order formula. A strong lower

bound on the expressive power of first order logic follows easily from Theorems 6.1 and 6.2.

COROLLARY 6.3. *For any k and any logical relation R on* $\mathbf{N}^k$,

$$\text{PARITY} \notin (\text{FO} + R)$$

Sipser also proved the following hierarchy result for bounded depth circuits:

THEOREM 6.4. [Si83]. *For all* $d > 1$ *there is a problem* S_d *accepted by polynomial size, depth d circuits; but not by polynomial size depth d circuits which have bounded fan-in at the bottom level.*

The problem S_d used by Sipser can be expressed by the first order formula

$$\exists x_1 \forall x_2 \ldots Q_d x_d L(x_1, \ldots, x_d)$$

over structures with a single relation L of arity d. A corollary of the above theorem is a very strong hierarchy result for first order logic:

COROLLARY 6.5. *For all* $d \geq 1$, *for all k, and for any logical relation R on* $\mathbf{N}^k$,

$$(\text{FO}, d+1 \textit{ alternations}) \not\subseteq (\text{FO} + R, d \textit{ alternations}).$$

7. Conclusions. We have shown that the important complexity classes, C, have corresponding languages, $\mathcal{L}$, such that C consists of exactly those properties expressible in $\mathcal{L}$. Thus questions of complexity can all be translated to expressibility issues. Separating complexity classes is equivalent to showing that certain of the above operators are more powerful than others. Efficient algorithms may be obtained simply by describing a problem in one of our languages. We have also demonstrated a basic connection between boolean circuits and first order logic.

Open questions and work to be done include the following:

1. More precise bounds on the arity of formulas needed to express properties of a given complexity are needed.
2. We have introduced first order translations as a new kind of reduction. Many open questions arise. It seems possible to prove that first order translations between certain problems cannot exist. Be careful however: showing that there is no first order translation from SAT to 1GAP would prove that NP $\neq$ L. (A tractable approach which we are presently pursuing is to prove lower bounds on the arity of projection translations between these problems.)
3. More knowledge is needed concerning the increase in expressibility gained by alternating applications of operators and negation. Let Σ_k^{TC} be first order logic in which k alternations of TC and negation are allowed, starting with TC. Thus for example, $\Sigma_1^{\text{TC}} = (\text{FO} + \text{ pos TC}) = \text{NL}$. We conjecture the following:

   (a): $\Sigma_1^{\text{TC}} \subsetneq \Sigma_2^{\text{TC}} \subsetneq \ldots$

   (b): $\Sigma_1^{\text{STC}} \subsetneq \Sigma_2^{\text{STC}} \subsetneq \ldots$.

   From a proof of any of the proper containments in (a) or (b) it would follow that $L \neq P$. Thus we would be satisfied with the more modest hierarchy results without s as a logical symbol:

   (c): $\Sigma_1^{\text{TC}}(\text{w.o. } s) \subsetneq \Sigma_2^{\text{TC}}(\text{w.o. } s) \subsetneq \ldots$.

   (d): $\Sigma_1^{\text{STC}}(\text{w.o. } s) \subsetneq \Sigma_2^{\text{STC}}(\text{w.o. } s) \subsetneq \ldots$.

4. We have shown that

$$\Sigma_*\text{Sym-}L \subseteq (\text{FO} + \text{STC}) \subseteq \text{BPL}.$$

It would be fascinating to know whether or not either of the above containments is proper.

5. Of course everyone would like to see a proof that some second order property is not expressible in first order plus least fixed point. This would imply that $P \neq \text{NP}$. It is very hard to prove strict hierarchy results for the languages described in this paper. However we feel that extensions to such results as Theorem 6.4 and Theorem 6.1 (together with Yao's strengthening of it) are steps in this direction.
6. Finally, we hope that attractive versions of the above languages will be developed for actual use as programming and/or database query languages.

Acknowledgments. My ideas for this paper have been clarified during many helpful discussions with colleagues. Thanks in particular to: Sam Buss, Steve Lucas, John Reif, Adi Shamir, and Mike Sipser. Thanks also to one of the referees for helpful suggestions on notation including the use of tables to describe formulas.

REFERENCES

[AHU74] A.V. AHO, J.E. HOPCROFT AND J.D. ULLMAN, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, Ma., 1974.

[AU79] A.V. AHO AND J.D. ULLMAN, *Universality of data retrieval languages*, Sixth Symp. on Principles of Programming Languages, 1979, pp. 110–117.

[Aj83] M. AJTAI, Σ_1^1 *formulae on finite structures*, Ann. Pure Appl. Logic, 24 (1983), pp. 1–18.

[AKLLR79] ALELIUNAS, KARP, LIPTON, LOVÁZ AND RACKOFF, *Random walks, universal traversal sequences, and the complexity of maze problems*, Proc. 20th IEEE Symp. on the Foundation of Computers, 1979, pp. 218–233.

[CH80b] A. CHANDRA AND D. HAREL, *Structure and complexity of relational queries*, 21st Symp. on Foundations of Computer Science, 1980, pp. 333–347. Also appeared in J. Comput. System Sci., 25 (1982), pp. 99–128.

[CKS81] A. CHANDRA, L. STOCKMEYER AND D. KOZEN, *Alternation*, J. Assoc. Comput. Mach., 28 (1981), pp. 114–133.

[CSV84] A. CHANDRA, L. STOCKMEYER AND U. VISHKIN, *Constant depth reducibility*, this Journal, 13 (1984), pp. 423–439.

[En72] H. ENDERTON, *A Mathematical Introduction to Logic*, Academic Press, New York, 1972.

[Fa74] R. FAGIN, *Generalized first-order spectra and polynomial-time recognizable sets*, in Complexity of Computation, R. Karp, ed., Society for Industrial and Applied Mathematics-American Mathematical Society Proc., 7, 1974, pp. 27–41.

[FSS81] M. FURST, J. B. SAXE AND M. SIPSER, *Parity, circuits, and the polynomial-time hierarchy*, Proc. 22nd IEEE Symp. on the Foundation of Computers, 1981, pp. 260–270.

[Go77] L. GOLDSCHLAGER, *The monotone and planar circuit value problems are log space complete for P*, SIGACT News, 9 (1977).

[Gr84] E. GRANDJEAN, *The spectra of first-order sentences and computational complexity*, this Journal, 13 (1984), pp. 356–373.

[HIM78] J. HARTMANIS, N. IMMERMAN AND S. MAHANEY, *One-way log tape reductions*, Proc. 19th IEEE Symp. on the Foundation of Computers, 1978, pp. 65–72.

[Im81] N. IMMERMAN, *Number of quantifiers is better than number of tape cells*, J. Comput. System Sci., 22 (1981), pp. 65–72.

[Im82a] ———, *Upper and lower bounds for first order expressibility*, J. Comput System Sci., 25 (1982), pp. 76–98.

[Im82b] ———, *Relational queries computable in polynomial time*, Proc. 14th Ann. ACM Symp. on the Theory of Computing, 1982, pp. 147–152.

[Im83] ———, *Languages which capture complexity classes*, Proc. 15th Ann. ACM Symp. on the Theory of Computing, 1983, pp. 347–354.

[LP82] H. LEWIS AND C. H. PAPADIMITRIOU, *Symmetric space bounded computation*, Theoret. Comp. Sci., 19 (1982), pp. 161–188.

[LG77] L. LOVÁSZ AND P. GÁCS, *Some remarks on generalized spectra*, Zeitchr. f. math, Logik und Grundlagen d. Math., 23 (1977), pp. 547–554.

[Ly82] J. LYNCH, *Complexity classes and theories of finite models*, Math. Systems Theory, 15 (1982), pp. 127–144.

[Re84] J. REIF, *Symmetric complementation*, J. Assoc. Comput. Mach., 31 (1984), pp. 404–421.

[Sa73] W. SAVITCH, *Maze recognizing automata and nondeterministic tape complexity*, J. Comput. System Sci., 7 (1973), pp. 389–403.

[Si83] M. SIPSER, *Borel sets and circuit complexity*, 15th Symp. on Theory of Computation, 1983, pp. 61–69.

[St77] L. STOCKMEYER, *The polynomial-time hierarchy*, Theoret. Comput. Sci., 3 (1977), pp. 1–22.

[SV85] S. SKYUM AND L. G. VALIANT, *A complexity theory based on boolean algeba*, J. Assoc. Comput. Mach., 32 (1985), pp. 484–502.

[SV84] L. STOCKMEYER AND U. VISHKIN, *Simulation of parallel random access machines by circuits*, this Journal, 13 (1984), pp. 409–422.

[Tu82] G. TURÁN, *On the definability of properties of finite graphs*, to appear.

[Va82] L. G. VALIANT, *Reducibility by algebraic projections*, L'Enseignement mathématique, T. XXVIII (1982), pp. 253–268.

[Va82] M. VARDI, *Complexity of relational query languages*, 14th Symp. on Theory of Computation, 1982, pp. 137–146.

[Ya85] A. CHI-CHIH YAO, *Separating the polynomial-time hierarchy by oracles*, 26th IEEE Symp. on Foundations of Comput. Sci., 1985, pp. 1–10.