Languages That Capture Complexity Classes*

Neil Immerman†

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.

*SIAM J. of Computing 16:4 (1987), 760-778. A preliminary version of this paper appeared as [17].

†Research supported by an NSF postdoctoral fellowship. Current address: Computer Science Dept., UMass, Amherst, MA 01003.

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).

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:

$$L \subseteq NL \subseteq \Sigma_* L \subseteq P \subseteq NP \subseteq \Sigma_* P$$

Here L is deterministic logspace and NL is nondeterministic logspace. $\Sigma_* L = \bigcup_{k=1}^{\infty} \Sigma_k L$ is the logspace hierarchy. $\Sigma_* P = \bigcup_{k=1}^{\infty} \Sigma_k P$ is the polynomial time hierarchy. Most knowledgable 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:

$$GAP = \{G | \exists \text{ a path in } G \text{ from } v_0 \text{ to } v_{n-1}\}$$

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:

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

Theorem 1.2 (HIM78) . *1GAP is one-way logspace complete for L.*

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

$$UGAP = \{G | G \text{ undirected and } \exists \text{ a path in } G \text{ from } v_0 \text{ to } v_{n-1}\}$$

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 (AKLL79) . *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 [LP80] 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. Let Sym-L be the class of problems accepted by symmetric logspace machines.

Theorem 1.4 (LP80) . *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).

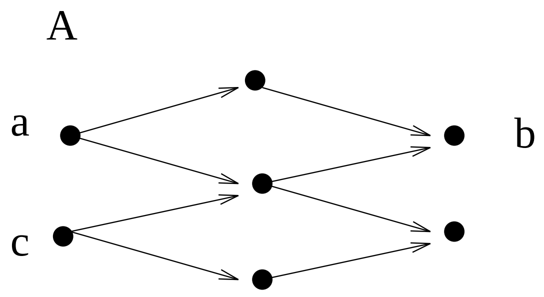

Figure 1: An alternating graph with two universal nodes: a, c.

Theorem 1.5 (Re84) .

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

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:

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 Figure 1.1 where APATH(a,b) holds, but APATH(c,b) does not. Let

$$AGAP = \{G | APATH_G(v_0, v_{n-1})\}$$

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]

[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].

2 First Order Logic

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

A finite *structure* with *vocabulary* $\tau = \langle \underline{R}_1...\underline{R}_k, \underline{c}_1...\underline{c}_r \rangle$ is a tuple, $S = \langle \{0, 1, \ldots, n-1\}, R_1...R_k, c_1...c_r \rangle$, consisting of a universe $U = \{0, \ldots, n-1\}$ and relations $R_1...R_k$ on U corresponding to the relation symbols $\underline{R}_1...\underline{R}_k$ of τ, and constants $c_1...c_r$ from U corresponding to the constant symbols $\underline{c}_1...\underline{c}_r$ from τ.

For example, if $\tau_0 = \langle \underline{E}(\cdot,\cdot) \rangle$ consists of a single binary relation symbol then a structure $G = \langle \{0...n-1\}, E \rangle$ with vocabulary τ_0 is a graph on n vertices. Similarly if $\tau_1 = \langle \underline{M}(\cdot) \rangle$ consists of a single monadic relation symbol then a structure $S = \langle \{0...n-1\}, M \rangle$ with vocabulary τ_1 is a binary string of length n.

If τ is a vocabulary, let

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

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 s is a logical relation symbol denoting the successor relation. We will also assume that *all structures under consideration have at least two elements.* Furthermore the constant symbols 0 and m 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: $=, s, 0, m$, using logical connectives: $\wedge, \vee, \neg$, variables: $x, y, z, ...$, and quantifiers: $\forall, \exists$.

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

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

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

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

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.*

[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 φ to the less appealing form: $(\forall x)(\forall y)(x = y \wedge \varphi') \vee (\exists x)(\exists y)(x \neq y \wedge \varphi(x/0, y/m))$.

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 $\varphi(x_1...x_k, y_1...y_k)$ be any formula. It represents a binary relation on k-tuples. We add to our language the operator TC where $TC[\varphi]$ denotes the reflexive, transitive closure of the relation φ. Let $(FO + TC)$ be the set of properties expressible using first order logic plus the operator TC. Let $(FO + \text{pos } TC)$ 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 $GAP \in (FO + \text{pos } TC)$:

$$GAP \equiv TC[E(x, y)](0, m)$$

Theorem 3.1 *$NL = (FO + pos\ TC)$*

Proof It is easy to see that NL contains $(FO + \text{pos } TC)$. If $A(x, y)$ can be checked in NL, then so can $TC[A(x, y)]$: 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 $(FO + TC)$ such that

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

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, $\tau = \langle \underline{E}(\cdot, \cdot) \rangle$, 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: $\langle q, r_1, r_2, w_1, h_1...w_k, h_k \rangle$. Here q codes M's state and variables r_1, r_2 code the input head position. Note that 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 j^{th} 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 j^{th} bit of w_i. Let $ON(w, j)$ 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. $PLUS(x, y, z) \equiv$ "$x + y = z$"
2. $ON(w, j)$

Proof Using s we can say that there is an edge from $\langle x, y\rangle$ to $\langle u, v\rangle$ if $u = x - 1$ and $v = y + 1$:

$$\alpha(x, y, u, v) \equiv \big(s(u, x) \wedge s(y, v)\big)$$

Using transitive closure we then get:

$$PLUS(x, y, z) \equiv TC[\alpha](x, y, 0, z)$$

Now define β as follows:

$$\beta(w_1, j_1, w_2, j_2) \equiv \big(\exists z(PLUS(w_2, w_2, z) \wedge (w_1 = z \vee s(z, w_1))) \wedge s(j_2, j_1)\big)$$

Let $ODD(z)$ abbreviate $\exists x \exists y(PLUS(x, x, y) \wedge s(y, z))$. It follows that

$$ON(w, j) \equiv \exists z\big(ODD(z) \wedge TC[\beta](w, j, z, 0)\big)$$

∎

Once we can tell what the work head is looking at we can write the predicate $NEXT(ID_a, ID_b)$ meaning that ID_b follows from ID_a in one move of M.[3] (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 $PATH(ID_a, ID_b)$ meaning that there is a computation of M starting at ID_a and leading to ID_b. Let $\psi = PATH(ID_i, ID_f)$ where ID_i and ID_f are M's initial and final ID's respectively. Then $\psi \in (FO{+}\text{pos } TC)$ and an input structure, G, of the correct vocabulary satisfies ψ if and only if M accepts G. This proves Theorem 3.1. ∎

The sentence $\psi = PATH(ID_i, ID_f)$ 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 $TC[\varphi](\bar{0}, \bar{m})$. 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*

$$TC[\psi(u_1, \ldots, u_k, v_1, \ldots, v_k)](\bar{0}, \bar{m})$$

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

[3] This is a standard constructions. See [Fa74, Im81, Im82a, Im82b].

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, v be variables not occuring in φ. Then $\varphi \leftrightarrow TC[\varphi(u,v)](0,m)$.

2. $\varphi \equiv TC[\psi(\bar{x},\bar{y})](\bar{q},\bar{r})$. We wish to replace $\bar{q},\bar{r}$ with $\bar{0},\bar{m}$. Put

$$\begin{aligned}\rho(s_1,t_1,\bar{x},s_2,t_2,\bar{y}) \equiv\ & \left[s_1 = 0 \wedge t_1 = 0 \wedge \bar{x} = \bar{0} \wedge s_2 = 0 \wedge t_2 = m \wedge \bar{y} = \bar{q}\right] \\ & \vee \left[s_1 = 0 \wedge t_1 = m \wedge s_2 = 0 \wedge t_2 = m \wedge \psi(\bar{x},\bar{y})\right] \\ & \vee \left[s_1 = 0 \wedge t_1 = m \wedge \bar{x} = \bar{r} \wedge s_2 = t_2 = m \wedge \bar{y} = \bar{m}\right]\end{aligned}$$

The variables t, s split the ρ-path in three stages: ($st = 00$): Set $\bar{x}$ to $\bar{q}$ and go to next stage. ($st = 0m$): Take a ψ step and stay in this stage. When you reach $\bar{r}$ go to next stage. ($st = mm$): Set $\bar{x} = \bar{m}$ and stop. Thus as desired:

$$\varphi \leftrightarrow TC[\rho(s_1,t_1,\bar{x},s_2,t_2,\bar{y})](\bar{0},\bar{m}) \ .$$

3. $\varphi \equiv \exists x TC[\alpha(\bar{u},\bar{v};x)](\bar{0},\bar{m})$. Here the notation means that the transitive closure is taken over the relation $\alpha(\bar{u},\bar{v})$ and the variable x occurs free in α. Put

$$\begin{aligned}\chi(\bar{u},x_1,\bar{v},x_2) \equiv\ & \left[\bar{u} = \bar{0} \wedge x_1 = 0 \wedge \bar{v} = \bar{0}\right] \\ \vee\ & \left[\alpha(\bar{u},\bar{v};x_1) \wedge x_1 = x_2\right] \vee \left[\bar{u} = \bar{m} \wedge \bar{v} = \bar{m} \wedge x_2 = m\right]\end{aligned}$$

Notice that χ allows a guess of x on the first step and thus,

$$\varphi \leftrightarrow TC[\chi(\bar{u},x_1,\bar{v},x_2)](\bar{0},\bar{m}) \ .$$

4. $\varphi \equiv \forall x TC[\alpha(\bar{u},\bar{v};x)](\bar{0},\bar{m})$. In this case we simulate the $\forall x$ by searching through all x's in order using s. I'm not sure whether the result holds if s is not available. Put

$$\nu(\bar{u},x_1,\bar{v},x_2) \equiv \left[\bar{u} \neq \bar{m} \wedge \alpha(\bar{u},\bar{v};x_1) \wedge x_1 = x_2\right] \vee \left[\bar{u} = \bar{m} \wedge \bar{v} = \bar{0} \wedge s(x_1,x_2)\right]$$

Thus $\quad \varphi \leftrightarrow TC[\nu(\bar{u},x_1,\bar{v},x_2)](\bar{0},\bar{m})$.

5. $\varphi \equiv TC\left[TC[\psi(\bar{x},\bar{y};\bar{u},\bar{v})](\bar{0},\bar{m})\right](\bar{0},\bar{m})$. In this case the formula ψ has free variables $\bar{x},\bar{y},\bar{u},\bar{v}$. The inner transitive closure is on $\bar{x},\bar{y}$, treating the other variables as parameters. The outer transitive closure is on $\bar{u},\bar{v}$. We combine these two TC's into a single transitive closure on δ defined as follows:

$$\begin{aligned}\delta(\bar{x},\bar{u}_1,\bar{v}_1,\bar{y},\bar{u}_2,\bar{v}_2) \ \equiv\ & \left[\bar{x} = \bar{y} = \bar{0} \wedge \bar{u}_1 = \bar{v}_1 = \bar{u}_2 = \bar{0}\right] \\ & \vee \left[\bar{x} \neq \bar{m} \wedge \bar{u}_1 \neq \bar{v}_1 \wedge \bar{u}_1 = \bar{u}_2 \wedge \bar{v}_1 = \bar{v}_2 \wedge \psi(\bar{x},\bar{y};\bar{u}_1,\bar{v}_1)\right] \\ & \vee \left[\bar{x} = \bar{m} \wedge \bar{y} = \bar{0} \wedge \bar{u}_2 = \bar{v}_1\right]\end{aligned}$$

We claim that $\varphi \leftrightarrow TC[\delta(\bar{x}, \bar{u}_1, \bar{v}_1, \bar{y}, \bar{u}_2, \bar{v}_2)](\bar{0}, \bar{m})$. This holds because a δ path consists exactly of a series of $\psi(\cdot, \cdot; u, v)$ paths from $\bar{0}$ to $\bar{m}$ with u, v fixed. At the end of any such path we know that $TC[\psi(x, y; u, v)](\bar{0}, \bar{m})$ holds. The δ path may now appropriately step from $(\bar{m}, \bar{u}, \bar{v})$ to $(\bar{0}, \bar{v}, \bar{w})$, i.e. reach v and begin trying to move from v to w.

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 $\tau_2 = \langle \underline{R}_1 \ldots \underline{R}_r \rangle$ and $\underline{R}_i$ is an a_i-ary relation symbol. Let k be a constant. An interpretation, σ, of $\mathcal{L}(\tau_2)$ in $\mathcal{L}(\tau_1)$ is a sequence of $r+1$ formulas:*

$$U(x_1 \ldots x_k), \qquad \Sigma_i(x_{11} \ldots x_{1k} \ldots x_{a_i k}) \quad i = 1 \ldots r$$

from $\mathcal{L}(\tau_1)$ such that the sentence $\exists x_1 \ldots x_k(U(\bar{x}))$ is valid.

Such an interpretation σ translates any structure $G \in STRUCT[\tau_1]$ to a structure $\sigma(G) \in STRUCT[\tau_2]$ defined as follows:

$$\begin{array}{ll} \text{Universe}(\sigma(G)) & = \{\langle x_1, \ldots, x_k \rangle \in G^k \mid G \models U(\bar{x})\} \\ R_i^{\sigma(G)} & = \{\langle \bar{x}_1, \ldots, \bar{x}_{a_i} \rangle \mid G \models \Sigma_i(\bar{x}_1, \ldots, \bar{x}_{a_i})\} \end{array}$$

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: $S \subset STRUCT[\tau_1]$ and $T \subset STRUCT[\tau_2]$, 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 $S \subseteq STRUCT[\tau_1]$ be any problem in NL. By theorems 3.1 and 3.3 there is a constant k and a formula $\gamma(x_1, \ldots, x_k, y_1, \ldots, y_k) \in \mathcal{L}(\tau_1)$ such that given $\mathcal{A} \in STRUCT[\tau_1]$ we have

$$\mathcal{A} \in S \Leftrightarrow \mathcal{A} \models TC[\gamma](\bar{0}, \bar{m}) \quad .$$

Let $\tau_2 = \langle \underline{E}(\cdot, \cdot) \rangle$. Let $\sigma = \langle U, \Sigma \rangle$ be an interpretation of $\mathcal{L}(\tau_2)$ in $\mathcal{L}(\tau_1)$ where $U(\bar{x}) \equiv (x_1 = x_1)$, $\Sigma(\bar{x}, \bar{y}) \equiv \gamma(\bar{x}, \bar{y})$. Then σ is a first order translation of S to GAP. ∎

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 $\varphi \in (FO+ \text{ pos } TC)$. Then φ is equivalent to a formula of the form $TC[\gamma](\bar{0}, \bar{m})$ where γ is a quantifier free formula in disjunctive normal form:*

$$\gamma \equiv \bigvee_{i=1}^{I} \alpha_i \wedge \beta_i \qquad \textit{where:}$$

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 $\sigma = \langle U, \Sigma_1, \ldots, \Sigma_j \rangle$ a *projection translation* if each of its formulas is in the form of γ in the above theorem. It follows that

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

Note that projection translations are a uniform version of the projection reductions of [SV85].

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

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

Proof Let M be a $\Sigma_k L$ turing machine and let $\mathcal{A}$ be an input structure to M. Let the predicates $EPATH_M(\bar{x}, \bar{y})$ (resp. $APATH_M(\bar{x}, \bar{y})$) 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:

$$\begin{aligned}(\exists ID_1 \forall ID_2 \exists ID_3 \quad & \ldots Q_k ID_k)[EPATH(ID_i, ID_1) \wedge \\ & [\neg APATH(ID_1, ID_2) \vee [EPATH(ID_2, ID_3) \wedge \ldots \wedge ID_k = ID_f] \ldots]\end{aligned}$$

∎

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

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 $\varphi(\overline{x}, \overline{y})$ is a first order relation on k-tuples, then $STC(\varphi)$ is the symmetric, transitive closure of φ.

$$STC(\varphi) \equiv TC[\varphi(x, y) \vee \varphi(y, x)]$$

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

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 $\varphi(\overline{x}, \overline{y})$ let

$$\varphi_d(\overline{x}, \overline{y}) \equiv \varphi(\overline{x}, \overline{y}) \wedge [(\forall \overline{z}) \neg \varphi(\overline{x}, \overline{z}) \vee (\overline{y} = \overline{z})]$$

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

$$DTC(\varphi) \equiv TC(\varphi_d)$$

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

Theorem 4.2

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

To prove Theorem 4.1 (resp. 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 $TC[\gamma](\bar{s},\bar{t})$ where for every tuple $\bar{x}$ there is at most one $\bar{y}$ such that $\gamma(\bar{x},\bar{y})$ and furthermore there is no $\bar{y}$ such that $\gamma(\bar{t},\bar{y})$. It follows that

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

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 $NEXT(ID_a, ID_b)$ 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 $\varphi \equiv \forall x STC[\alpha(\bar{u},\bar{v};x)](\bar{0},\bar{m})$. We let

$$\nu((\bar{u},x_1,\bar{v},x_2) \equiv \left[\bar{u} \neq \bar{m} \wedge \alpha(\bar{u},\bar{v};x_1) \wedge x_1 = x_2\right] \vee \left[\bar{u} = \bar{m} \wedge \bar{v} = \bar{0} \wedge s(x_1,x_2)\right]$$

and find that

$$\varphi \leftrightarrow STC[\nu(\bar{u},x_1,\bar{v},x_2)](\bar{0}0,\bar{m}m) \, .$$

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

$$(p_0 \times 0),\ \langle(\bar{m},0),(\bar{0},1)\rangle,\ (p_1 \times 1),\ \ldots,\ \langle(\bar{m},m-1),(\bar{0},m)\rangle,\ (p_m \times m)$$

is a ν path from $\bar{0}0$ to $\bar{m}m$. Conversely, any minimal ν path from $\bar{0}0$ to $\bar{m}m$ must include the edges,

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

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, $STC[\alpha(\bar{u},\bar{v};x)[\bar{0}0,\bar{m}m]]$, 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: $\varphi \equiv \exists x DTC[\alpha(\bar{u},\bar{v};x)](\bar{0},\bar{m})$. Here instead of letting the path finder guess the correct x we force the path to try all x's and go to $\bar{m}$ 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 $arity(\bar{z}_i) = arity(\bar{w}_i) = arity(\bar{u})$. 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 '$s(\bar{z}_1, \bar{z}_2)$' to mean that $\bar{z}_2$ is the successor of $\bar{z}_1$ in the lexicographical ordering induced by the successor relation s. Let

$$\begin{aligned}
&\chi'(\bar{u}, \bar{z}_1, \bar{w}_1, x_1, \bar{v}, \bar{z}_2, \bar{w}_2, x_2) \equiv \big[\bar{u} = \bar{v} = \bar{z}_2 = \bar{w}_2 = \bar{m} \wedge x_2 = m\big] \\
&\quad \vee \big[\bar{u} \neq \bar{m} \wedge s(x_1, x_2) \wedge \bar{z}_1 = \bar{m} \wedge \bar{v} = \bar{z}_2 = \bar{w}_2 = \bar{0}\big] \\
&\quad \vee \big[\bar{u} \neq \bar{m} \wedge s(x_1, x_2) \wedge \bar{z}_1 \neq \bar{m} \wedge \bar{w}_1 = \bar{m} \wedge \bar{v} = \bar{z}_2 = \bar{w}_2 = \bar{0} \wedge \neg\alpha(\bar{u}, \bar{w}_1; x_1)\big] \\
&\quad \vee \big[\bar{u} \neq \bar{m} \wedge x_2 = x_1 \wedge s(\bar{z}_1, \bar{z}_2) \wedge \bar{w}_2 = \bar{0} \wedge \bar{v} = \bar{w}_1 \wedge \alpha(\bar{u}, \bar{w}_1; x_1)\big] \\
&\quad \vee \big[\bar{u} \neq \bar{m} \wedge x_2 = x_1 \wedge \bar{z}_2 = \bar{z}_1 \neq \bar{m} \wedge s(\bar{w}_1, \bar{w}_2) \wedge \bar{v} = \bar{u} \wedge \neg\alpha(\bar{u}, \bar{w}_1; x_1)\big]
\end{aligned}$$

Then

$$\varphi \leftrightarrow DTC[\chi](\bar{0}, \bar{m}) .$$

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 $\Sigma_* Sym\text{-}L = (FO + STC)$ but we suspect that equality does *not* hold. We can prove the following:

Theorem 4.3

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

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 $\alpha(\bar{x}, \bar{y})$ is in BPL, we can test if $\alpha(\bar{x}, \bar{y})$ holds with exponentially low probablity of error. Then we can take a random α walk and use Theorem 1.3 to see that $STC[\alpha(\bar{x}, \bar{y})]$ 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:

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

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:

$$LFP(\varphi) \equiv min\{R | \varphi(R) = R\}$$

It is well known that $LFP(\varphi)$ exists and is computable in polynomial time in the size of the structure involved.

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:*

$$\begin{aligned}
\Gamma(R)[x, y] \equiv \quad & (x = y) \vee \Big[(\exists z)(E(x, z) \wedge R(z, y)) \\
& \wedge \big(\neg A(x) \vee (\forall z)(\neg E(x, z) \vee R(z, y))\big)\Big]
\end{aligned}$$

It is easy to see that

$$LFP(\Gamma) = APATH$$

and in fact:

Theorem 4.5 (Im82a,Va82) . $P = (FO + LFP)$

Instead of using LFP we introduce a variant of it that is more in keeping with DTC,STC, and TC. Suppose that $\varphi(\bar{x}, \bar{y})$ and $\alpha(\bar{x})$ are given formulas where $arity(x) = arity(y) = k$. 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 $\varphi(\bar{x}, \bar{y}), \alpha(\bar{x})$ is $APATH_{G_{\varphi,\alpha}}$. We can define ATC more precisely in terms of LFP as follows. Refering to example 4.4, let

$$\begin{aligned} \Gamma(\varphi, \alpha, R)[x, y] \equiv \quad & (x = y) \vee \Big[(\exists z)(\varphi(x, z) \wedge R(z, y)) \\ & \wedge \big(\neg\alpha(x) \vee (\forall z)(\neg\varphi(x, z) \vee R(z, y))\big)\Big] \end{aligned}$$

Definition 4.6 $ATC[\varphi, \alpha] \equiv LFP(\Gamma(\varphi, \alpha))$

We conclude this section with

Theorem 4.7

1. *$P = (FO + ATC)$*
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 ASPACE[log n] = P, [CKS81]. The construction of $NEXT(ID_a, ID_b)$ 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 ASPACE[log n] machine M accepts a structure $\mathcal{A}$ iff $\mathcal{A} \models ATC(NEXT, \alpha)(ID_i, ID_f)$.

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) . $NP = (2^{nd}O \text{ existential})$

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 expresible in second order logic.

Theorem 5.2 (St77) . $\Sigma_* P = (2^{nd}O)$

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:

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

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

Corollary 5.4

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

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

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

where $w_1, \ldots, w_{c-1}$ 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

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

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

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

where ψ is first order.

In particular $\psi \equiv (\exists x_1 \ldots Q_a x_a) M(\bar{R}, \bar{x})$ where M is a constant size quantifier free formula. It follows that we can test in $\Sigma_{c+a} TIME[n^k]$ 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 $|\langle \mathcal{A}, R_1, \ldots, R_c \rangle|$. ∎

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 $TC(\varphi)$, $STC(\varphi)$, $DTC(\varphi)$ 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,...*

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

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, $X_1 \ldots X_c$, and to write the first order sentence $\varphi_M(X_1 \ldots X_c, Y_1 \ldots Y_c)$ 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,...*

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

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 on.

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 inwhich 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 $\tau = \langle M(\cdot)\rangle$ consisting of a single monadic predicate. Thus to a first order formula an input of size n is a structure $\mathcal{A} = \langle\{0\dots n-1\}, M_{\mathcal{A}}\rangle$. To a boolean circuit the same input is the string of n boolean variables $a_1 \dots a_n$, where $(a_i = 1) \Leftrightarrow i \in M_{\mathcal{A}}$.

Suppose that (1) holds and that the circuits $C_1, C_2, \dots$ 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, $z_1 z_2 \dots z_k$. In this way each 'or' gate v at the bottom level has a label $z_{11}z_{12}\dots z_{1k}z_{21}\dots z_{dk}$. Each such gate in C_n is a disjunction $c_1 \vee \dots \vee c_k$ of literals: a_i or $\bar{a}_i$. Define the relation Q as follows: $Q(m, z_{11}, z_{12}, \dots, z_{dk}, y, j)$ holds iff j=0 (resp. j=m) and literal a_y (resp. $\bar{a}_y$) occurs in the gate with label $z_{11}z_{12}\dots z_{dk}$ in C_n. I know that this is a mouthful, but the point is that the logical relation Q codes all the circuits $C_1, C_2, \dots$. Define the sentence

$$\begin{aligned}\psi \equiv\ & \forall z_{11}\dots z_{1k}\exists z_{21}\dots z_{2k}\dots\forall z_{d1}\dots z_{dk}\exists y\big(Q(m,\bar{z},y,0)\wedge M(y)\\ & \vee R(m,\bar{z},y,m)\wedge\neg M(y)\big)\end{aligned}$$

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. Define the relation R as follows: For all $m, \bar{z}$ if only $y_1, \dots y_{k-c}$ satisfy $Q(m,\bar{z},y,0) \vee Q(m,\bar{z},y,m)$, then choose $w_1, \dots w_c$ distinct from these y_i's (note that we may assume that $k < n$). Let

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

and put

$$\begin{aligned}\alpha \equiv \quad & \forall y_1 \ldots \forall y_k \Big(\big[y_1 \neq y_2 \wedge y_1 \neq y_3 \wedge \ldots \wedge y_{k-1} \neq y_k \quad \wedge (R(m,\bar{z},y_1,0) \vee R(m,\bar{z},y_1,m)) \\ & \wedge \ldots \wedge (R(m,\bar{z},y_k,0) \vee R(m,\bar{z},y_k,m))\big] \; \rightarrow \big[(\neg R(m,\bar{z},y_1,0) \wedge \neg M(y_1)) \\ & \vee(\neg R(m,\bar{z},y_1,m) \wedge M(y_1)) \vee \ldots \vee (\neg R(m,\bar{z},y_k,m) \wedge M(y_k))\big]\Big)\end{aligned}$$

Finally let

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

Then φ is as required.

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

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

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 independant 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 $\mathbf{N}^k$,*

$$PARITY \notin (FO + R)$$

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 $\mathbf{N}^k$,*

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

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 $NP \neq L$.

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, $\Sigma_1^{TC} = (FO+ \text{ pos } TC) = NL$. We conjecture the following:

   (a): $\Sigma_1^{TC} \overset{\subset}{\neq} \Sigma_2^{TC} \overset{\subset}{\neq} \dots$

   (b): $\Sigma_1^{STC} \overset{\subset}{\neq} \Sigma_2^{STC} \overset{\subset}{\neq} \dots$

   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): $\Sigma_1^{TC}(\text{w.o. } s) \overset{\subset}{\neq} \Sigma_2^{TC}(\text{w.o. } s) \overset{\subset}{\neq} \dots$

   (d): $\Sigma_1^{STC}(\text{w.o. } s) \overset{\subset}{\neq} \Sigma_2^{STC}(\text{w.o. } s) \overset{\subset}{\neq} \dots$

4. We have shown that

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

   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 $P \neq NP$.

6. Finally, we hope that attractive versions of the above languages will be developed for actual use as programming and/or database query languages.

Acknowledgements 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.

References

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms,* Addison- Wesley, 1974.

[2] A.V. Aho and J.D. Ullman, "Universality of Data Retrieval Languages," *Sixth Symp. on Principles of Programming Languages,* 1979, (110-117).

[3] M. Ajtai, "Σ_1^1 Formulae on Finite Structures," *Annals of Pure and Applied Logic* **24**, 1983, (1-48).

[4] Aleliunas, Karp, Lipton, Lováz, and Rackoff, "Random Walks, Universal Traversal Sequences, and the Complexity of Maze Problems," *20th IEEE FOCS Symp.*, 1979, (218-233).

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

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

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

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

[9] 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).

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

[11] L. Goldschlager, "The Monotone and Planar Circuit Value Problems are Log Space Complete for P," *SIGACT News,* **9**, No. 2, 1977.

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

[13] Juris Hartmanis, Neil Immerman, and Stephen Mahaney, "One-Way Log Tape Reductions," *19th IEEE FOCS Symposium,* 1978, (65-72).

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

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

[16] Neil Immerman, "Relational Queries Computable in Polynomial Time," *14th ACM STOC Symp.,* (1982), 147-152.

[17] Neil Immerman, "Languages Which Capture Complexity Classes," *15th ACM STOC Symp.,* (1983) 347-354.

[18] Harry Lewis and Christos H. Papadimitriou, "Symmetric Space Bounded Computation," *Theoret. Comp. Sci.,* bf 19, No. 2, 1982, (161-188).

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

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

[21] John Reif, "Symmetric Complementation," *JACM* **31**, No. 2, April, 1984, (401-421).

[22] W. Savitch, "Maze Recognizing Automata and Nondeterministic Tape Complexity," *JCSS* **7**, 1973, (389-403).

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

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

[25] S. Skyum and L.G. Valiant, "A Complexity Theory Based on Boolean Algebra," *JACM,* 32, No. 2, April, 1985, (484-502).

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

[27] G. Turán, "On the Definability of Properties of Finite Graphs."

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

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

[30] Andrew Chi-Chih Yao, "Separating the Polynomial-Time Hierarchy by Oracles," *26th IEEE Symp. on Foundations of Comp. Sci.*, 1985, 1-10.