Languages Which Capture Complexity Classes

(Preliminary Report)

Neil Immerman *

Department of Mathematics
Tufts University
Medford,Mass. 02155

and

Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Mass.. 02139

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 graph property is in polynomial time if and only if it is expressible in the language of first order graph theory together with a least fixed point operator. As another example, a group theoretic property is in the logspace hierarchy if and only if it is expressible in the language of first order group theory together with a 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 and least fixed point 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).

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

*Research supported by NSF grant MCS81-05754.

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 and 4 introduce the languages under consideration. Section 5 considers the relationship between first order logic and polynomial 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:

$$GAP = \{ \langle G, a, b \rangle \mid \exists \text{ a path in } G \text{ from } a \text{ to } b \}$$

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:

$1GAP =$
$\{ \langle G, a, b \rangle \mid G$ has outdegree 1 and $\exists$ a path in G from a to $b \}$

Theorem 1.2 [HIM78]: 1GAP is one-way logspace complete for L.

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

$UGAP =$
$\{ \langle G, a, b \rangle \mid G$ undirected and $\exists$ a path in G from a to $b \}$

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) $> 3/4$, while if $w \notin S$ then Prob(M accepts w) $< 1/4$. 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 [Re82] 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.

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. $A \subseteq V$ 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:

(i): APATH(x,x)

(ii): If x is existential and for some child z of x APATH(z,y) holds, then APATH(x,y).

(iii): If x is universal, has at least one child, and all of its children, z, satisfy APATH(z,y), then APATH(x,y).

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

$$AGAP = \{ \langle G, a, b \rangle \mid APATH_G(a,b) \}$$

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 1.5 [Im81]: AGAP is logspace complete for P.

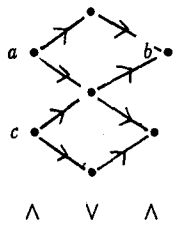

Figure 1.1: An alternating graph.

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, $S = \langle \{1...n\}, R_1 ... R_k, c_1 ... c_r \rangle$, consisting of a universe $U = \{1...n\}$ 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 $G = \langle \{1...n\}, E \rangle$ 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 $S = \langle \{1...n\}, M \rangle$ with vocabulary τ_1 is a binary string of length n.

If τ is a vocabulary, let

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

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.

The *first order language* $L(\tau)$ is the set of formulas built up from the relation symbols of τ and the logical relation symbols: $=$, $<$, using logical connectives: $\wedge, \vee, \neg$, variables: $x, y, z, ...$, and quantifiers: $\forall, \exists$. The relation symbol $<$ refers to the usual ordering on the universe of integers $1...n$, and the quantifiers range over this universe. We will say more in Section 3 about the need for $<$.

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

$$MOD(\varphi) = \{ G \in STRUCT(\tau) \mid G \text{ satisfies } \varphi \}$$

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

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

The following theorem is well known, [AU79, Im81]; but a new proof of the strictness of the containment follows from Corollary 5.3.

Theorem 2.1: FO is strictly contained in L.

3. First Order Logic With Closure Operators

In this section we add operators to first order logic in order to form languages in which interesting properties of finite structures are expressible. First we consider a transitive closure operator (TC).

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.

Theorem 3.1:

(a): $GAP \in (FO + TC)$

(b): $NL = (FO + \text{pos } TC)$

(c): $\Sigma_* L = (FO + TC)$

proof:

(a): The GAP property is easily expressed using TC:

$$GAP \equiv TC[E(x,y)](a,b)$$

(b): 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

$$S = \{\, G \in STRUCT(\tau) \mid G \text{ satisfies } \psi \,\}$$

We will sketch the construction of ψ. The first idea is that an instantaneous description (ID) of M can be coded with finitely many variables ranging from 1 to n.

Example 3.1: Suppose 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: $\langle q, r_1, r_2, w_1, h_1 \ldots w_k, h_k \rangle$. Here q codes M's state and variables r_1, r_2 code the input head position. Note that 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 $w_1 \ldots w_k$ code the $k \cdot log(n)$ bits of M's work tape. One h_i is equal to 2^j where the work head is pointing to the j^{th} bit of w_i; the rest of the h_i's are 1.

The second idea is that using TC we can compute the j^{th} bit of w_i. Let $ON(w,h)$ mean that $h = 2^j$ for some j and bit j of w is on. Starting with $<$ we can use TC to express addition and multiplication and thus tell if a certain bit in a variable is on:

Lemma 3.1: The following predicates are expressible in $(FO + \text{pos } TC)$.

(a): $PLUS(x,y,z) \equiv$ "$x + y = z$"

(b): $TIMES(x,y,z) \equiv$ "$x \cdot y = z$"

(c): $ON(w,h)$

proof of (a): Using $<$ it is easy to express "1" and the successor function, "s". We can then say that there is an edge from (x,y) to (u,v) if $u = x - 1$ and $v = y + 1$:

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

Using transitive closure we then get:

$$PLUS(x,y,z) \equiv TC[EDGE](s(x), y, 1, z)$$

∎

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. (Note that we make a crucial use here of the ordering, $<$, 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.) 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(b). Therem 3.1(c) follows by closing each side of (b) with negation. ∎

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 and these may easily be merged into one. Thus for each problem C in NL there is a 2k-ary first order formula φ, and two first order definable k-tuples ID_i, ID_f such that a structure G is a member of C iff G satisfies $TC[\varphi](ID_i, ID_f)$. This suggests the following:

Definition: 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 τ_1 as τ_2 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 $L(\tau_1)$.

Thus σ translates each structure $G \in STRUCT[\tau_1]$ to a structure $\sigma(G) \in STRUCT[\tau_2]$. $\sigma(G)$ has universe the set of k-tuples from the universe of G which satisfy U. The relations, $R_1 \ldots R_r$, on this universe are given by the formulas $\Sigma_1 \ldots \Sigma_r$.

See [En72] for a discussion of interpretations between theories.

Given two problems: $S \subseteq STRUCT[\tau_1]$ and $T \subseteq STRUCT[\tau_2]$, a *first order translation* of S to T is an interpretation, σ, of τ_1 as τ_2 such that

$$G \in S \quad \text{if and only if} \quad \sigma(G) \in T$$

The above discussion proves:

Corollary 3.2: GAP is complete for NL via first order translations.

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 3.3:

(a): $UGAP \in (FO + STC)$

(b): $Sym\text{-}L = (FO + \text{pos } STC)$

(c): $\Sigma_* Sym\text{-}L = (FO + STC)$

(d): $UGAP$ is complete via first order translations for $Sym\text{-}L$

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 $(FO + \text{pos } DTC)$ is closed under negation. Thus:

Theorem 3.4:

(a): $1GAP \in (FO + DTC)$

(b): $L = (FO + DTC)$

(c): $1GAP$ is complete for L via first order translations.

The last 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 $R_1 \subseteq R_2$ implies $\varphi(R_1) \subseteq \varphi(R_2)$. 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 3.2: 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:

$$\psi(R)[x,y] \equiv (x = y) \vee [(\exists z)(E(x,z) \wedge R(z,y)) \wedge (\neg A(x) \vee (\forall z)(\neg E(x,z) \vee R(z,y)))]$$

It is easy to see that

$$LFP(\psi) = APATH$$

A proof of the following first appeared in [Im82a].

Theorem 3.5: $P = (FO + LFP)$

Recalling that $P = ASPACE[log(n)]$ the proof of Theorem 3.1 may be modified slightly to show that any P-time property may be translated into a single instance of the AGAP property.

Corollary 3.6: AGAP is complete for P via first order translations.

4. 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 4.1 [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 "<" 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 4.1 under negation gives us that a problem is in the polynomial time hierarchy iff it is expresible in second order

logic.

Corollary 4.2 [St77]: $\Sigma_* P = (2^{n^d}O)$

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:

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

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 4.3 [Ly82]:

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

Corollary 4.4:

(a): $\Sigma_r TIME[n^k] \subseteq (2^{n^d}O$ with $+$, r alterntions, arity k)

(b): $\Sigma_* TIME[n^k] = (2^{n^d}O$ with $+$, arity k)

Note that in the above results the relation "+" need only be added when k is 1, 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 4.5: $PSPACE = (2^{n^d}O + TC) = (2^{n^d}O + STC) = (2^{n^d}O + DTC)$

proof sketch: 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, $S_1 \ldots X_c$, and to write the first order sentence $\varphi(X_1 \ldots X_c, Y_1 \ldots Y_c)$ meaning that ID $\overline{X}$ follows from $\overline{X}$ in one move of some turing machine M. One aplication of the appropriate transitive closure operator expresses an entire computation. ∎

In fact we have proved the following:

Theorem 4.6: For k=1,2,...

(a): $DSPACE[n^k] = (2^{n^d}O$ arity k, $+ DTC)$

(b): $Sym - SPACE[n^k] = (2^{n^d}O$ arity k, $+$ pos $STC)$

(c): $NSPACE[n^k] = (2^{n^d}O$ arity k, $+$ pos $TC)$

Finally we can add a least fixed point operator on monotone second order operators. That is we add the ability to define super relations on relations by induction.

Theorem 4.7: For k = 1,2,...

$$DTIME[2^{O[n^k]}] = (2^{n^d}O \text{ arity k } + LFP)$$

5. Lower Bounds

A recent lower bound by Furst, Saxe and Sipser has interesting consequences for us. The PARITY problem is to determine whether the n inputs to a boolean circuit include an even number which are on.

Theorem 5.1 [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 5.2: Given a problems, S, the following are equivalent:

(a): S is recognized by a sequence of bounded depth, polynomial size circuits.

(b): There is a fixed constant, k, and a sequence of first order sentences, $\varphi_1, \varphi_2 \ldots$ such that φ_n expresses S for structures of size n; and, φ_n has exactly k quantifiers. The sentence φ_n belongs to a first order language which includes the new logical relations $I_1 \ldots I_n$ where I_j is a monadic relation true exactly of point j of the universe.

proof:

(b) implies (a) because k quantifiers are easy to simulate with a k level circuit of fan in n, giving total size n^k. The quantifier free part can be expressed as an $O[n^k]$ disjunction of bounded conjunctions because everything is bounded except the n^k possible ways that the k chosen variables may satisfy $I_1 \ldots I_n$.

(a) implies (b): For simplicity think of an input structure with a single monadic relation, M, i.e. a sequence of n boolean inputs. We can say that the j^{th} input is on:

$$(\exists x) I_j(x) \wedge M(x)$$

Suppose that some and-gate, A, has inputs $G_1 \ldots G_n$. Assume inductively that each G_i is expressible by a first order sentence φ_i with $k-1$ quantifiers:

$$\varphi_i \equiv (\exists x_2)(\forall x_3)\ldots(Q_k x_k)\alpha_i(x_2 \ldots x_k)$$

Then A can be expressed by the sentence:

$$\Phi \equiv (\forall x_1)(I_1(x_1) \wedge \varphi_1) \vee \cdots \vee (I_n(x_1) \wedge \varphi_n)$$

which may be then simplified to:

$$\Phi \equiv (\forall x_1)(\exists x_2)\ldots(Q_k x_k)$$
$$(I_1(x_1) \wedge \alpha_1(x_2 \ldots x_k)) \vee \cdots \vee (I_n(x_1) \wedge \alpha_n(x_2 \ldots x_k))$$

In this way an inductive proof shows that any polynomial size circuit of bounded depth, k, whose gate fan in is limited to n^c can be expressed in the above first order language using $k \cdot c$ quantifiers. ∎

The new relation symbols, $I_1 \ldots I_n$ were added to capture the power of the circuits which may examine any of their inputs. These relation symbols allow the φ_n's to express any relation on the universe not involving M. For example, φ_{17} can express the property "$x + y = z$" by saying:

$$(I_1(x) \wedge I_1(y) \wedge I_2(z)) \vee \cdots \vee (I_9(x) \wedge I_8(y) \wedge I_{17}(z))$$

A first order theory is any set of first order axioms. Equivalently we may think of it as a set of structures with certain given relations satisfying prescribed rules. Thus:

Corollary 5.3: Parity of a new relation M is not expressible in any first order theory.

We may also think of our elements $\{1 \ldots n\}$ as the power set of the smaller universe $\{1 \ldots log(n)\}$. Using the language of Corollary 5.2 we can easily express the relation "$a \in b$" meaning that the a^{th} bit of the binary representation of b is one. Now $\underline{M}$ may be thought of as a monadic relation symbol on subsets.

Corollary 5.4: There is no second order monadic sentence $\varphi(\underline{M})$ using the new symbol $\underline{M}$, referring to an arbitrary monadic relation on subsets, such that $\varphi(\underline{M})$ says, "M is true of an even number of subsets."

Of course PARITY can be expressed using transitive closure or any other operator allowing a linear traversal of the input. Thus the above corollaries also show that TC is not first order definable, nor is TC of an expression involving a new relation symbol second order monadic definable. Furthermore we can think of Corollary 5.4 in terms of oracles. Since no φ expresses parity for all M's we can build a single relation M that foils all φ's. (Each φ will be wrong at some finite point, so with a finite initial segment of M we can kill the first r φ's.) Recalling Corollary 4.4 we conclude:

Corollary 5.5: There is an oracle M such that

$$PARITY(M) \in (DSPACE^M(n) - \Sigma_* TIME^M(n))$$

Of Course Corollary 5.5 would be more interesting if we could either remove the oracle from it or change the n to "any polynomial in n."

Conclusions

We have shown that the important complexity classes, C, have corresponding languages, L, such that C consists of exactly those properties expressible in 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:

- More precise bounds on the arity of formulas needed to express properties of a given complexity are needed.
- We have introduced first order translations as a new kind of reduction. Many open questions arise. The relative power of these as compared with other reductions should be determined. 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 $NP \neq L$ These very neat reductions also suggest that it may now be plausible to ask such questions as, "What makes a particular second order existential sentence NP complete?"

- There are some open questions concerning separating classes with oracles. For example, proving that PARITY(M) $\notin (2^{n^d}O + M)$ would imply that there is an oracle separating the polynomial time hierarchy from PSPACE.
- More knowledge is needed concerning the increase in expressibility gained by alternating applications of operatores and negation. Let $(FO + k \bullet TC)$ be first order logic in which k alternations of TC and negation are allowed. Thus for example, $(FO + 1 \bullet TC) = (F) +$ pos $TC) = NL$. We conjecture the following:

  (a): $(FO + 1 \bullet TC) \subsetneq (FO + 2 \bullet TC) \subsetneq \ldots$

  (b): $(FO + 1 \bullet STC) \subsetneq (FO + 2 \bullet STC) \subsetneq \ldots$

  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 $<$ as a logical symbol:

  (c): $(FO$ w.o.$< + 1 \bullet TC) \subsetneq (FO$ w.o.$< + 2 \bullet TC) \subsetneq \ldots$

  (d): $(FO$ w.o.$< + 1 \bullet STC) \subsetneq (FO$ w.o.$< + 2 \bullet STC) \subsetneq \ldots$
- 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 $P \neq NP$.
- 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 friends and colleagues over the past year and a half. I am grateful to Adi Shamir, John Reif, Yuri Gurevich, and Mike Sipser.

Bibliographical Note

Gács and Lovász have studied "elementary reductions", a notion similar to first order translations. They showed, [LG77], that SAT is NP complete via elementary reductions.

Vardi, [Va82], has obtained some nice results on the complexity of languages with added operators including least fixed point and transitive closure.

References

[AHU74] A.V. Aho, J.E. Hopcroft, J.D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1974.

[AU79] A.V. Aho, J.D. Ullman, "Universality of Data Retrieval Languages," *6th Symp. on Principles of Programming Languages*, 1979, pp. 110-117.

[AKLL79] Aleliunas, Karp, Lipton, Lovasz, Rackoff, "Random Walks, Universal Traversal Sequences, and the Complexity of Maze Problems," *20th IEEE FOCS Symposium*, Oct. 1979, pp. 218-223.

[CH80] A.K. Chandra, D. Harel, "Structure and Complexity of Relational Queries," *21st IEEE FOCS Symposium*, Oct. 1980, pp. 337-347.

[CKS81] A.K. Chandra, D.C. Kozen, L.J. Stockmeyer, "Alternation," *JACM* Vol. 28, No. 1, Jan. 1981, pp. 114-133.

[En72] H. Enderton, *A Mathematical Introduction to Logic*, Academic Press, 1972.

[Fa74] R. Fagin, "Generalized First-Order Spectra and Polynomial-Time Recognizable Sets," in *Complexity of Computation*, (ed. R. Karp), *SIAM -AMS Proc. 7*, 1974, pp. 27-41.

[FSS81] M. Furst, J.B. Saxe, M. Sipser, "Parity, Circuits, and the Polynomial-Time Hierarchy," *22nd IEEE FOCS Symposium*, Oct. 1981, pp. 260-270.

[Gr83] E. Grandjean, "The Spectra of First-Order Sentences and Computational Complexity," to appear.

[HIM78] J. Hartmanis, N. Immerman, S. Mahaney, "One-Way Log Tape Reductions," *19th IEEE FOCS Symposium*, 1978, pp. 65-72.

[Im81] N. Immerman, "Number of Quantifiers is Better than Number of Tape Cells," *JCSS* Vol. 22, No. 3, June 1981, pp. 384-406.

[Im82a] N. Immerman, "Relational Queries Computable in Polynomial Time," *14th ACM SIGACT Symposium*, May, 1982, pp. 147-152.

[Im82b] N. Immerman, "Upper and Lower Bounds for First Order Expressibility," *JCSS, Vol. 25,* No. 1, August, 1982.

[LP80] H.R. Lewis, C.H. Papadimitriou, "Symmetric Space Bounded Computation," *ICALP80.*

[LG77] L. Lovász, P. Gács, "Some Remarks on Generalized Spectra," *Zeitschr. f. math. Logik und Grundlagen d. Math,* Bd. 23, pp. 547-554, 1977.

[Ly82] J. Lynch, "Complexity Classes and Theories of Finite Models," *Math. Sys. Theory 15*, 1982, pp. 127-144.

[Re82] J. Reif, "Symmetric Complementation," *14th ACM SIGACT Symposium*, May, 1982, pp. 201-223.

[Sa73] W. Savitch, "Maze Recognizing Automata and Nondeterministic Tape Complexity," *JCSS* Vol. 7, 1973, pp. 389-403.

[Si83] M. Sipser, "Borel Sets and Circuit Complexity," *this volume.*

[St77] L. Stockmeyer, "The Polynomial-Time Hierarchy," *TCS* Vol. 3, 1977. pp. 1-22.

[Tu82] G. Turán, "On the Definability of Properties of Finite Graphs," to appear.

[Va82] M. Vardi, "Complexity of Relational Query Languages," *14th ACM SIGACT Sympsium*, May 1982, pp.137-146.