NUMBER OF QUANTIFIERS IS BETTER THAN NUMBER OF TAPE CELLS

by

Neil Immerman

TR 80-410

Department of Computer Science
Cornell University
Ithaca, New York 14853

February 1, 1980

Abstract

We introduce a new complexity measure, $QN[f(n)]$, which clocks the size of sentences from predicate calculus needed to express a given property. Techniques from logic are used to prove sharp lower bounds in the measure. These results demonstrate space requirements for computations and may provide techniques for separating Time and Space complexity classes because we show that:

$$NSPACE[f(n)] \subseteq QN[f(n)^2/\log(n)] \subseteq DSPACE[f(n)^2].$$

Introduction and Summary:

For the purpose of analyzing the time and space requirements of computations, we introduce a new complexity measure. Most measures count how much of some computational resource (e.g. time or memory space) is needed to check whether an input has a certain property, C. Instead, we examine the number of quantifiers needed to express C in first order predicate logic.

The result is Quantifier Number (QN) a bonafide complexity measure which is not based on a machine model. It turns out that QN agrees closely with space complexity, and yet it does not distinguish between deterministic and nondeterministic space. Thus we have a model whose lower bounds translate directly into lower bounds for space, and yet is sufficiently different to allow new methods and ideas to be brought to bear. In particular there are well established methods in logic to decide what can and cannot be said in various languages. These techniques provide lower bounds having nothing to do with complete sets or diagonalization.

We hope to convince even the skeptical reader that it makes more intuitive sense to try to prove a lower bound (by induction, say) on the number of quantifiers needed to express a certain property, than on the number of Turing machine tape cells needed to check if the property holds for a given input. (Hence our title.)

This paper grew out of work by Fagin (see [Fag74]). He proved the following:

<u>THEOREM</u> (<u>Fagin</u>): A set, S, of structures is in NP if and only if there exists a sentence, F, with the following properties:

1. $F = (\exists P_1)\cdots(\exists P_k)H(P_1,\ldots,P_k)$, where $P_1,\ldots,P_k$ are predicates and H is a first order sentence.
2. Any structure, G, is in S iff G satisfies F.

Thus a property is in NP just if it is expressible by a second order existential sentence. (3-colorability of graphs is a good example of such a property.)

It is difficult to show lower bounds for the expressiblity of second order sentences. Instead we examine first order sentences which, we found, mimic computations much more closely. Considering graph problems, for example, the length of the shortest sentence which says, "G is connected," grows as the logarithm of the size of G. It is not a coincidence that this is also the space needed by a Turing machine to test if G is connected.

To study this growth of sentences we introduce the complexity measure QN which will be defined in Section 1. Informally, a set, S, of structures is in QN[f(n)] if membership in S for those structures of size less than or equal to n can be expressed by a sentence with f(n) quantifiers.

The sentences mentioned above are written in the language of the given structures. For example if we are dealing with graph problems then the quantifiers range over the vertices and there is a single relation symbol, E(-,-), representing the edge relation. It seems that this language suffices to describe "natural" problems on graphs, but to simulate an arbitrary Turing Machine computation we must give the language access to an ordering of the universe. We let $QN^S[f(n)]$ be the family of properties expressible with f(n) quantifiers in a language that includes Suc(-,-), a successor relation. We can now show that NSPACE[f(n)] is contained in $QN^S[f(n)^2/\log(n)]$.

We will say that C is in QN[f(n)] only if there is a <u>uniform</u> sequence of sentences expressing C. The uniformity (in the sense of Borodin, see [Bor77]) allows us to prove $QN[G(n)] \subseteq DSPACE[G(n)\log(n)]$, and thus:

$$NSPACE[f(n)] \subseteq QN^S[f(n)^2/\log(n)] \subseteq DSPACE[f(n)^2].$$

Note that our lower bounds will not consider the uniformity; they may be interpreted in the strongest possible sense. When we show that C is not in QN[f(n)] we mean that <u>no sentence with n quantifiers</u> expresses C for structures of size n.

The <u>quantifier rank</u> of a sentence T is the depth of nesting of quantifiers in T. Thus a sentence with n quantifiers has at most quantifier rank n. In Section 2 we consider a two person game with which we prove lower bounds for quantifier rank. An <u>Ehrenfeucht game</u> is played on a pair of structures G,H of the same type. Player I chooses points to show that G and H are different, while Player II matches these points, trying to keep the structures looking the same. A theorem due to Fraisse and Ehrenfeucht says that Player II has a winning strategy for the n move game if and only if G and H agree on all sentences of quantifier rank n. The original treatment of these games appears in [Ehr61] and [Fra54].

Ehrenfeucht games provde a lower bound technique for QN as follows. Given some property, C, we find structures G and H of size n such that G satisfies C but H does not. We then show that Player II has a winning strategy for the f(n) move game on G and H. It follows that G and H agree on all sentences of quantifier rank f(n) and thus in particular no sentence with f(n) quantifiers can express the property C. Thus we have shown that C is not in QN[f(n)].

These combinatorial games provide very sharp lower bounds. We show for example that while quantifier rank log(n) suffices to express the graph property, "There is a path from point a to point b," quantifier rank log(n) -2 is insufficient!

In Section 3 we present a more sophisticated Ehrenfeucht game argument. We show that without successor quantifier rank $(\log n)^k$ is insufficient to describe a set recognizable in polynomial time. If our proof went through for the language with successor we would have shown that PTIME is not contained in $\bigcup_{k=1}^{\infty} \text{SPACE}[(\log n)^k]$.

Making the above result go through with successor is a major technical problem which we have not (yet) solved. For one thing we show that quantifier rank is no longer the right thing to check. Any property whatsoever of graphs of size n can be expressed by a sentence with 2^{n^2} quantifiers but quantifier rank only log(n) in a language with successor.

To make matters worse two ordered graphs G and H satisfy all of the same formulas with 3(log n) quantifiers only if they are identical. This is as expected because G and H are indistinguishable to all log space Turing machines only if they are identical. The proof is the same in both cases: the machine or the short sentence can check if vertex 3 is connected to vertex 17. G and H agree on all such tests only if they are identical (i.e. the map from vertex i in G to vertex i in H is an isomorphism.)

We leave this paper open ended by proposing a few possible techniques for adding successor to the above result and thus proving that $P \notin \bigcup_{k=1}^{\infty} QN^S[(\log n)^k]$. The most hopeful one at present is a modification of Ehrenfeucht games such that Player I wins the k move game if and only if a given property is expressible with k quantifiers and Suc(-,-). This new game is combinatorially much more complex than the Ehrenfeucht game and so we are by no means proficient at playing it. And yet we wanted to present, as a point of departure for future research, what may become a viable technique for proving space lower bounds for problems which are not complete for a certain amount of space.

SECTION 0: Review of some notions from logic.

A *structure*, $S=\langle U,c_1^U,\ldots,c_k^U,P_1^U,\ldots,P_n^U\rangle$, consists of a universe, U, certain constants, $c_1^U,\ldots,c_k^U$ from U, and, certain relations, $P_1^U,\ldots,P_n^U$, on U.

A *similarity type*, $T=\langle c_1,\ldots,c_k,P_1,\ldots,P_n\rangle$, is a sequence of constant symbols and relation symbols.

As an example let G be a directed graph with two specified points s and d. Thus, $G = \langle V,E^G,s^G,d^G\rangle$ is a structure of type $T_g = \langle E,s,d\rangle$, where V is the set of vertices of G, and E^G is G's edge relation.

If T is any type then L(T), the language of T, is the set of all sentences built up from the symbols of T using &, or, , ->, =, variables x,y,z,...; and the quantifiers $(\exists x)$ and (x).

A sentence, F, in L(T) is given meaning by a structure, S, of type T as follows: The symbols from T are interpreted by the constants and relations in S. The quantifiers in F range over the elements of the universe of S.

For example, let $A = (x)(x{=}d \text{ or } (\exists y)E(x,y))$. A is in $L(T_g)$. Furthermore, G satisfies A (in symbols, $G \models A$) iff each vertex of G except d^G has an edge coming out of it. Henceforth we will omit the superscript G for the sake of readability.

The *quantifier rank* of sentence F, (qr[F]), is the depth of nesting of quanitifiers in F. Inductively,

$$qr[(x)B] = qr[(\exists x)B] = qr[B]+1$$
$$qr[B\&C] = qr[B \text{ or } C] = \max(qr[B],qr[C]).$$

For example, for $A= (x)[((\exists y)\ P(x,y))\ \&\ ((z)(w)Q(x,z) \text{ or } L(z,w))]$, $qr[A] = 3$.

The number of elements in the universe of S is abbreviated $|S|$. For graphs $|G|$ is the number of vertices of G.

SECTION 1: The quantifier measure.

We are now ready to make our principal definiton. We say that a set, C, of structures of type T is in QN[h(n)] if there exists a sequence of sentences $\{F_i | i=1,2,...\}$ from L(T), and a constant, k, such that:

a. For all structures, G, of type T, if $|G| \leq n$, then:

$$G \text{ is in } C \quad \text{<->} \quad G \models F_n.$$

b. F_n has $\leq k(h(n))$ quantifiers.

c. The map $f{:}n \text{->} F_n$ is generable by a DSPACE[h(n)] Turing machine.

Thus C is in QN[h(n)] if there is a uniform sequence of sentences whose n^{th} member has O(h(n)) quantifiers and expresses the membership property of C for structures of size n. Our condition (c) is analagous to Borodin's notion of a problem's circuit depth inwhich he considers uniform sequences of boolean circuits (see [Bor77]).

As an example, let GAP be the set of directed graphs, G, with two distinguished points, s and d, such that there is a path in G from s to d. GAP is a set of structures of type T_g = <E(-,-),s,d>. Membership in GAP is known to be complete for NSPACE[log(n)]. (See [Sav73].)

Theorem 1: GAP is in QN[log(n)].

proof: We must assert that there is a path of length at most n from s to d. We define by induction the sentences $P_k(x,y)$. $P_k(x,y)$ says that there is a path of length at most k from x to y.

$$P_1(x,y) \quad = \quad (x=y) \text{ or } E(x,y)$$
$$P_{2k}(x,y) \quad = \quad (\exists z)(P_k(x,z) \;\&\; P_k(z,y))$$

The sentence P_n has quantifier rank n and 2^n existential quantifiers. Using a familiar trick, (see [FiRa74] or [Sav70]), we can add universal quantifiers and reduce the total number of quantifiers to 3(log(n)):

$$A_1(x,y) \quad = \quad P_1(x,y)$$
$$A_{2k}(x,y) \quad = \quad (\exists z)(u)(v)[(u=x \;\&\; v=z) \text{or} (u=z \;\&\; v=y)] \to A_k(u,v)$$

Thus, letting $F_n = A_n(s,d)$, we have shown that GAP is in QN[log(n)]. □

Although the complete problem GAP is in QN[log(n)], it is not true that NSPACE[log(n)] is contained in QN[log(n)]. As we show in Section 2, this fails in a rather spectacular way: the regular set, EVEN = {G | G has an even number of vertices}, is not in QN[log(n)].

To allow them to simulate Turing machines it suffices to give the sentences access to the numbering of the vertices which the machines already have. Thus we define below the measure QN^S which studies properties expressible with an arbitrary successor relation, Suc(-,-). Suc(x,y) means that y comes just after x in the numbering of the elements of the universe.

A similar Suc relation is discussed in [Sav73]. Savitch shows that his pebble automata cannot accept GAP without Suc. However, Theorem 1 suggests that our sentences do not need Suc to express "natural" graph problems.

Definition: We say that a set, C, of structures of type T is in QN^S[h(n)] if there exists a sequence of formulas $\{F_i | i=1,2,\ldots\}$ from L(T U Suc), and a constant k such that:

a. For all structures, G, of type T, with $|G| \leq n$, and for all binary relations Suc(-,-), if Suc(-,-) is a valid successor relation on the universe of G then:

$$(G \in C) \quad \leftrightarrow \quad \langle G, Suc(-,-)\rangle \models F_n \quad .$$

b. F_n has at most k(h(n)) quantifiers.

c. The map $f: n \to F_n$ is generable by a DSPACE[h(n)] Turing machine.

Thus a property, C, is in QN^S[h(n)] if there is a uniform sequence of h(n) quantifier sentences from L(T U {Suc}) which give the same answer for any successor relation and express C. In the author's thesis the gain in expressibility given by adding an arbitrary relation satisfying a certain condition is considered. Using this device for various conditions we can capture PTIME and other classes.

The following theorem shows that with the addition of Suc(-,-) Quantifier Number is closely related to SPACE:

__Theorem 2__: Let f(n) be any function such that $f(n) \geq \log(n)$. Then:

$$\mathrm{NSPACE}[f(n)] \subseteq \mathrm{QN}^S[\frac{f(n)^2}{\log(n)}] \subseteq \mathrm{DSPACE}[f(n)^2] .$$

__proof__: We sketch the proof of the first inclusion. The idea is that each element of the universe has a number from 1 to n, and so may be treated as log(n) bits. Thus a Turing machine instantaneous description (id) may be coded in O(f(n)/log(n)) variables. (A similar technique appears in [Sto77].) It is not hard to prove by induction that O[log(n)] quantifiers suffice to say, "Digit i of element x is 0." Thus with O[log(n)] quantifiers we can say "ID_2 follows from ID_1 in one step." Note that the crucial use of Suc is in saying, "Now the Turing machine moves its input head one space to the right."

The length of a computation may be $c^{f(n)}$ so, as in the proof of Theorem 1, we need O[f(n)] id's to state that such a path exists.

The second inclusion: We show that QN[g(n)] is contained in DSPACE[g(n)log(n)] for any function g(n). Given G of size n we can generate F_n in SPACE[g(n)]. Check the truth of a g(n) quantifier sentence in DSPACE[g(n)log(n)] as follows: Cycle through the sentence with all possible values of the quantified variables. If F_n is of the form (x)H(x) then we test the truth of $H(v_i)$ for each vertex v_i of G. Each variable requires log(n) bits and g(n) of them must be remembered at once. When all the variables in F_n have been replaced by constants its truth may be checked as we generate it with no additional space required. □

<u>SECTION 2</u>: Ehrenfeucht Games.

In this section we will employ Ehrenfeucht games to obtain lower bounds for the quantifier measure. These games are due to Fraisse and Ehrenfeuct. (See [Fra54] or [Ehr61] for discussion and proof of Theorem 3.) Two persons play the game on a pair of structures. Player I tries to demonstrate a difference between the two structures, while Player II tries to keep them looking the same. An example appears below, but first we give the definition and state the fundamental fact about these games.

Given two structures, G and H, of the same finite type, T, we define the n move game on G and H as follows:

Player I chooses an element of G or H and Player II chooses a corresponding element from the other one. This is repeated n times. At move i, g_i and h_i, elements of G and H respectively, are chosen.

We say that <u>Player II wins</u> if the map f which takes the constants from G to the constants from H, and maps g_i to h_i, is an isomorphism of the induced substructures. (That is f preserves all of the symbols of T. For example, if G and H are of type $T=T_g$, $E(s,g_1)$ holds in G just if $E(s,h_1)$ holds in H.)

We say that two structures of type T are <u>n-equivalent</u> if they satisfy all the sames sentences in L(T) of quantifier rank n. The fundamental fact about Ehrenfeucht games is:

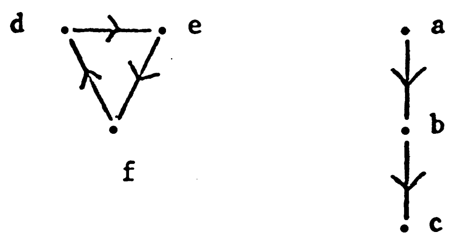
<u>Theorem 3(Fraisse,Ehrenfeucht)</u>: Player II has a winning strategy for the n move game on A,B, iff A is n-equivalent to B.

As an example, consider the graphs G and H of Figure 1. G has the property that each of its vertices has an edge leading to it, but this is not true of vertex a in H. Thus G and H disagree on the sentence, $S = (x)(\exists y)(E(y,x))$. By Theorem 3, Player I has a winning strategy for the game of length 2. Indeed, on the first move Player I chooses a. II must answer with a point from G, say d. Now I can pick f from G. II will lose because there is no point in H with an edge to a.

FIGURE 1:

	G	H
1.	d	a
2.	f	—

G H



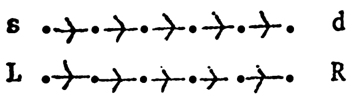
Recall that in Theorem 1 we showed an upper bound of QN[log(n)] for GAP. The following theorem proves that this is a lower bound as well. Note that we can express GAP in quantifier rank exactly log(n), and so the Ehrenfeucht game is a tool fine enough to decide expressibility up to an additive constant!

Theorem 4: GAP is not expressible in quantifier rank log(n) -2.

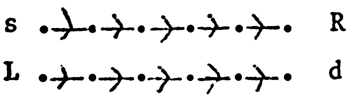
proof: Fix n>4 and let m=[(n-4)/2]. We construct the graphs A_m,B_m as follows: Each graph consists of two lines of m+2 vertices as in Figure 2. In both graphs s is the top left vertex; but, d is the top right vertex in A_m and the bottom right vertex in B_m. Thus A_m is in GAP, but B_m is not.

FIGURE 2:

A_4



B_4



We will now show that A_m is (log(n)-2)-equivalent to B_m. From this it follows that no sentence of quantifier rank log(n)-2 can express the property, "There is a path from s to d."

By Theorem 3 it suffices to show that Player II wins the log(m) move game on A_m, B_m. Indeed, the following is a winning strategy for II:

If Player I plays the i^{th} vertex in some row of A (or B), II will always answer with the i^{th} vertex of one of the rows in B (or A). The initial constraint is that the endpoints s,d,L,R are answered by the similarly labelled endpoints. With k moves to go, if Player I chooses vertex x within 2^k steps of an endpoint (or previously chosen vertex, a_i), then II must answer with a vertex on the same row as the corresponding endpoint (or b_i).

A proof by induction will show that if II follows the above strategy for log(m) moves, then a conflict (i.e. two points on different rows, both within 2^k steps) will never arise. Thus Player II wins the log(n)-2 move game. $\square$

Theorem 4 remains true for ordered graphs. The proof is similar, but the graphs require three rows each so that d is not the last vertex in B_m.

It is interesting to note that in the above case our measure does not distinguish between deterministic and nondeterministic space. The lower bound of O[log(n)] is shown for graphs with at most one edge leaving any vertex. The gap problem for such graphs, (called GAP1 and discussed in [HIM78] and [Jon75]), is in DSPACE[log(n)].

As promised we now show that $L(T_g)$, the language of graphs without Suc, is insufficient for describing all graph problems. Our counterexample consists of a totally disconnected graph. The same example could be built with connected graphs of unbounded degree. The idea is that the edge relation is of no use and so we must name all the points inorder to count them.

Proposition 5: EVEN, the set of graphs with an even number of vertices, is in DSPACE[log n], (in fact it's in DSPACE[0]), but is not in QN[log(n)], (in fact it's not in QN[h(n)] for any h(n) asympotically less than n).

proof: We already know by Theorem 2 that EVEN is in QN^S[log(n)]. To prove Proposition 5 let TD_n be the totally disconnected graph with n vertices. We show that TD_{n-1} is n-1 equivalent to TD_n. It follows that quantifier rank n is needed to express EVEN.

We only need to show that Player II wins the n-1 move game on TD_{n-1} and TD_n. Her* obvious winning strategy is to match a chosen vertex with any vertex from the other side subject to the condition that a point chosen twice will be answered with the same point both times. Since the edge relation is always false in both structures, the resulting sequences of points are isomorphic. □

The proposition above concerns itself with the difference between QN and QN^S. In the next section we will produce a more natural graph problem in P-TIME, which is not in $QN[\log(n)^k]$. The graphs there are connected and of bounded degree. We feel that the latter example

*Inorder to avoid the awkward construction "he/she" in complicated game arguments we adopt the convention that Player I is male and Player II is female.

concerns itself with time versus space.

SECTION 3: P-TIME and the QN measure.

Let an alternating graph be a directed acyclic graph whose vertices are marked "&" or "or". Suppose that a and b are vertices of alternating graph G, and a has edges to $x_1, \ldots, x_n$. We say that b is reachable from a iff:

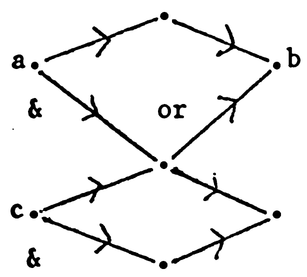
1. a = b; or,

2. a is marked "&", $n \geq 1$, and b is reachable from all the x_i's; or,

3. a is marked "or" and b is reachable from some x_i.

Note that if all vertices are marked "or" then this is the usual notion of reachability. (See Figure 3 where b is reachable from a, but not from c.) Note that we could generalize this definition to include infinite graphs or graphs with cycles by saying that "b reachable from a" is the smallest relation satisfying 1-3.

FIGURE 3: An Alternating Graph



Now define AGAP to be the set of alternating graphs inwhich d is reachable from s.

Proposition 6: AGAP is complete for polynomial time.

proof: To see if G is in AGAP, we start at d, and proceeding backwards mark all the points from which d is reachable.

A detailed proof of completeness is omitted; the idea is that AGAP is complete in a natural way for alternating log space, which is known to be equivalent to P-TIME. (See [ChSt76] or [Koz76].) Boolean circuit value problems which are very similar have previously been shown to be complete for P. See for example [Gol77]. □

We must now add the predicate A(x) meaning that vertex x is marked "&". Let T_{ag}=<E,A,s,d>, be the type of alternating graphs. Our next theorem shows that in $L(T_{ag})$ the polynomial time property AGAP is not expressible with quantifier rank $(\log n)^k$. If this went through with the addition of successor then we would have shown that P is not contained in SPACE[$(\log n)^k$].

<u>Theorem 7</u>: Let f(n) be any function that is asymptotically less than $2^{\sqrt{\log(n)}}$. Then AGAP is not in QN[f(n)]. In particular, AGAP is not in QN[$\log^k(n)$] for any k.

<u>proof</u>: For all sufficiently large m, we produce graphs G_m and H_m with the following properties:

1. $|G_m| = |H_m| = n$, and $n < m^{\log(m)}\log(m)$. Thus $\log(n) < \log(m)(\log(m)+1)$, and $2^{\sqrt{\log(n)+1}} < m$.

2. G_m is m-equivalent to H_m.

3. G_m is in AGAP, but H_m is not.

When these conditions are met we will have shown that anything less than quantifier rank $2^{\sqrt{\log(n)}}$ does not suffice to express the Alternating Graph Accessibility Problem without successor.

The first step is to introduce the building block out of which G_m and H_m will be constructed:

<u>Lemma 7a</u>: Let X be the alternating graph pictured in Figure 4. Then X has automorphisms f,g, and h, with the following properties:

1. f switches 3 & 4 and 1 & 2, leaving 5 & 6 fixed.

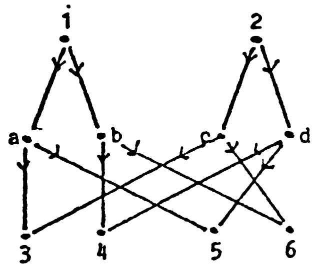
2. g switches 1 & 2 and 5 & 6, leaving 3 & 4 fixed.

3. h switches 3 & 4 and 5 & 6, leaving 1 & 2 fixed.

FIGURE 4: Switch X

or

and



proof: The idea is that when X is placed in our graphs each pair, 1,2, 3,4, 5,6, will consist of one point which can reach d and one which cannot. Think of points which can reach d as "true," and those which cannot as "false." Then, in symbolic notation:

1 = a or b = (3 & 5) or (4 & 6)
2 = c or d = (3 & 6) or (4 & 5)

The proof of the lemma is an easy computation. □

We will say that a pair u,v, is "off" if u is true and v is false. If u is false and v is true then the pair is "on." Thus, X is a switch whose top pair is on just if exactly one of its bottom pairs is on.

FIGURE 5: P_m (if s=A), Q_m (if s=B)

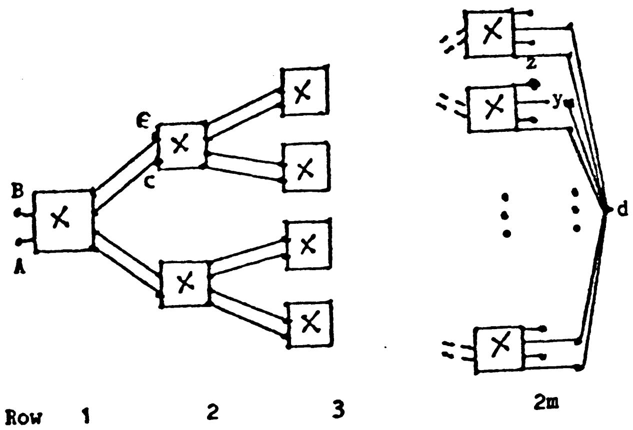


Figure 5 shows $2^{2m+1}-1$ copies of the switch X, arranged in a binary tree. Let P_m be the graph pictured in Figure 5, with s=A. Let Q_m be the same graph, but with s=B. Thus P_m is in AGAP while Q_m is not. However,

Lemma 7b: P_m is m-equivalent to Q_m.

proof: We will show that Player II wins the m length game on P_m and Q_m. One way to express the difference between P_m and Q_m is to say that they are the same except that the top pair in Q_m is switched. Another way of thinking of it is that in Q_m one of the bottom pairs, for example y,z, is switched. That is in P_m y is connected to d, but in Q_m z is connected to D. X has the property that switching one pair on the bottom will result in the top pair being switched.

The idea behind Player II's winning strategy is that the difference between P_m and Q_m could be removed by switching any of the 2^{2m} pairs on the bottom row. With only m moves, Player I cannot eliminate all of these possiblities.

To simplify the proof let us first consider a different game. Let T_{2m} be the binary tree of height 2m. This is a schematic version of P_m and Q_m where each point represents the switch, X, and each line represents a pair of lines.

We play a modified Ehrenfeucht game on T_{2m}, call it the on-off game. On each move of this new game, Player I picks a point and Player II must answer "on" or "off". Player II must also obey the rules that the top vertex, if chosen, is on, and any chosen vertex on the bottom is off. (Intuitively "off" corresponds to matching the top left vertex of the chosen switch in P_m to the same vertex in Q_m; "on" means matching it to the top right vertex.) We say that Player II wins if for any triple of chosen points, L,M,N, such that M and N are the two offspring of L, L is on iff exactly one of M and N is on. This rule captures the behavior of the switch X.

Lemma 7c: Suppose that each vertex in row r of T_n is labelled on or off. Then any 2^k-1 points on or below row r+k may be labelled in any self-consistent fashion and there will still be a labelling of the rest of the graph which generates row r.

proof: By induction on k. If k=1 then no matter which point is chosen we are free to label its sibling as we please inorder to give the desired label to its parent.

Inductively suppose that 2^k-1 points are labelled on or below row r+k. Let L be the set of left offspring in row r+1, R the set of right offspring. Clearly at most one of these sets, say L, has more than $2^{k-1}-1$ of its descendants labelled. Label all of the vertices in L in any consistent fashion. Now by induction we may label the points in R as we choose. Thus we may label row r as desired. □

It follows that Player II wins the 2m-move on-off game on T_{2m}. Her strategy is to answer "off" whenever possible. The lemma shows that she can never be forced to declare the n^{th} row on in an n-move game.

We can now play the original m-move Ehrenfeucht game as follows (see Figure 5): When Player I chooses a point, for example c in P_m, II moves according to the strategy for the on-off game. If the point corresponding to c's switch is declared "off", then II answers c, if "on", then d, the opposite point in the pair. If a point inside a switch is chosen then II may simulate the moves of the on-off game for the switch's two descendants, and move accordingly. This proves Lemma 7b. □

The final step of the proof is to introduce the graph $D_{\log m}$ to replace the binary tree in the above construction. $D_{\log m}$ has $m^{\log(m)}$ vertices but still has the property that no point in block k can be forced on before the k^{th} move. We define D_k below, algebraically, but please refer to Figures 6 and 7 which show D_2 and the first three blocks of D_3, respectively.

VERTICES(D_k) =
$\{<x_1,\ldots,x_k,r> \mid r=b*k+p,\ p<k, b<2^k, 0\le x_i\le b+1 \text{ for } 1\le i\le p \ \&\ 0\le x_i\le b \text{ for } p<i\le k\}$

EDGES(D_k) =
$\{ (<x_1,\ldots,x_k,r>, <x_1,\ldots,x_k,r+1>) \mid r<k*2^k\}$ U
$\{ (<x_1,\ldots,x_k,r>, <x_1,\ldots,x_{p-1},x_p+1,\ldots,x_k,r+1>) \mid r=k*b + p \}$

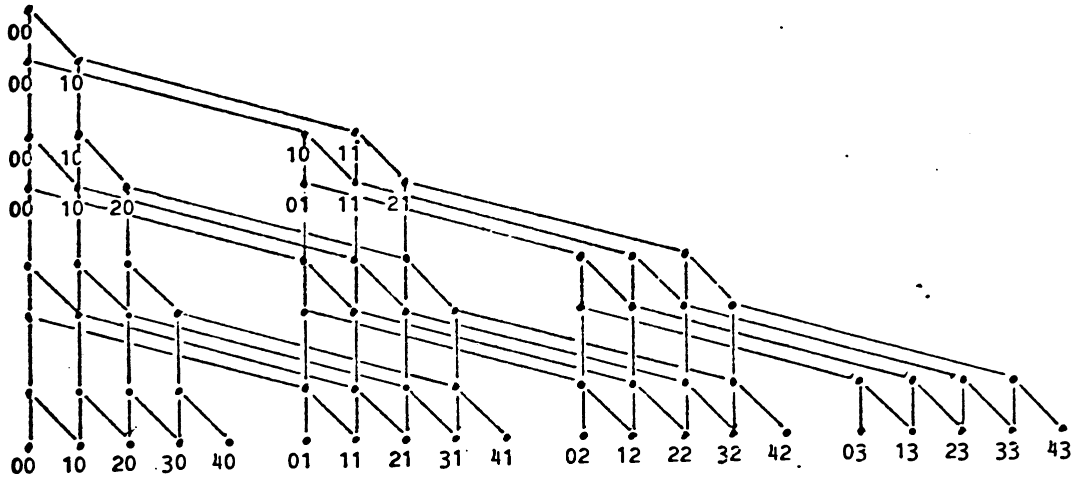
Thus the vertices are k dimensional vectors and each row stretches the range of one of these dimensions by one. These graphs have k degrees of freedom, allowing us to prove:

Lemma 7d: Suppose that row r of D_k is entirely labelled. Then any 2^k -1 points on or below row r+k may be labelled in any self-consistent fashion and there will still be a labelling of the rest of the graph which generates row r.

proof: By induction on k. If k=1 then we must show that any one point may be chosen in row r+1 without affecting row r. This is true because any configuration in row r is generated by a configuration in row r+1 and by its complement.

Suppose we have our lemma for k-1 and consider any labelling of row r in D_k. For convenience assume that row r is the bottom row of the j^{th} block. Thus the chosen 2^k -1 points are on the bottom row of the $j+1^{st}$ block or below. Note that the subgraph of D_k with fixed first

FIGURE 6: D_2



ROW

$0 = 0*2 + 0$

$1 = 0*2 + 1$

$2 = 1*2 + 0$

$3 = 1*2 + 1$

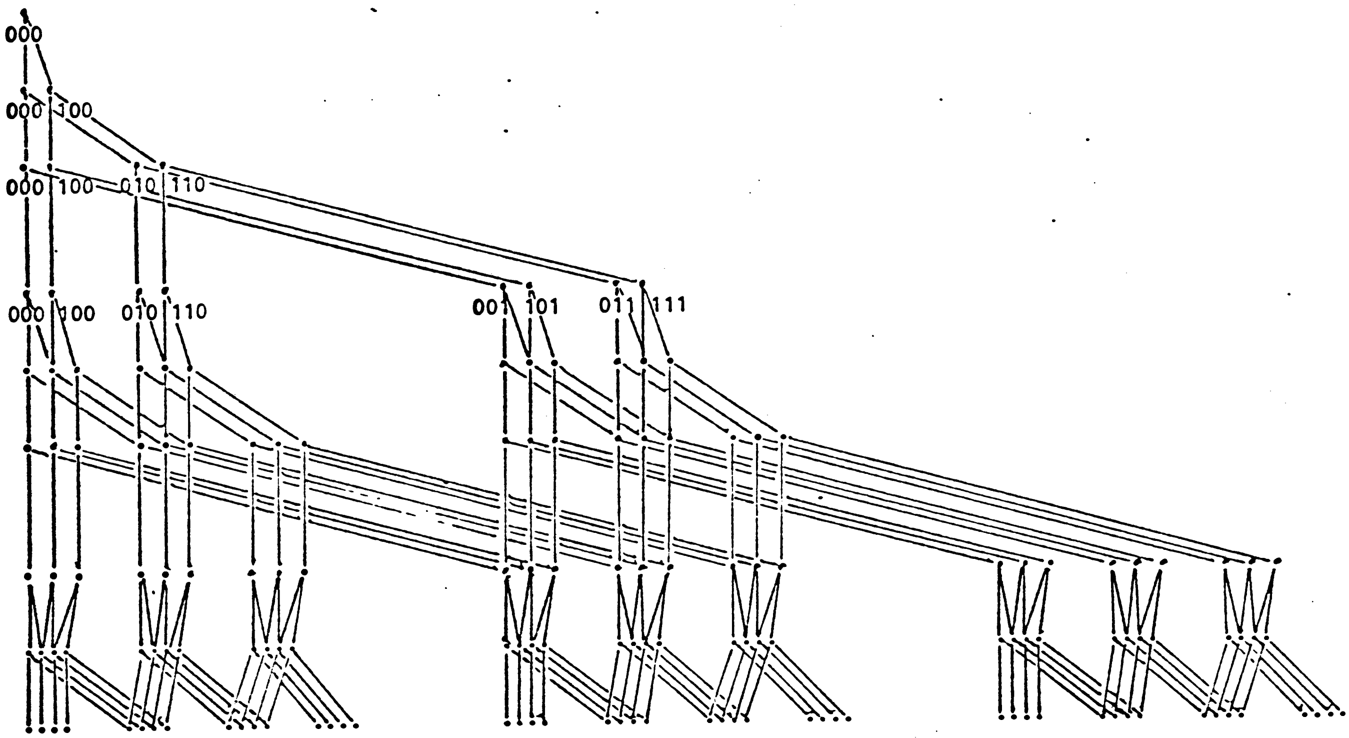
$4 = 2*2 + 0$

$5 = 2*2 + 1$

$6 = 3*2 + 0$

$7 = 3*2 + 1$

FIGURE 7: Three Blocks of D_3



coordinate i is a copy of D_{k-1}. Furthermore for at most one i_0 are there 2^{k-1} chosen vertices with first coordinate i_0. Label the i_0^{th} column of the j+1st block in any consistent fashion. Now by induction since less than 2^{k-1} -1 vertices in any other column are chosen, we can set the rest of row r+1 as we please. Thus as in the case of D_1 we have control of j of the j+1 points in each group. Thus we may generate row r as desired. □

From Lemma 7d it follows that Player II wins the 2^k move on-off game on D_k. Let G_m and H_m be the graphs arising from $D_{\log(2m)}$ by replacing vertices by the switch X, just as P_m and Q_m arose from T_{2m}. As before we let s be the top left point of G_m and the top right point of H_m. Thus G_m is in AGAP, H_m is not in AGAP, and G_m is m-equivalent to H_m. This proves Theorem 7. □

Theorem 7 does not go through if we add "Suc". In the log(n) move game on numbered graphs if Player I chooses vertex i in A, then II must respond with vertex i in B. If Player II answers differently, then in the remaining moves Player I can keep cutting the successor path from the initial point to vertex i in half, thus exposing that this path is not the same length on the left as on the right. Clearly if G and H are not identical graphs there must be a pair of indiced i,j such that there is an edge from v_i to v_j in one of the graphs but not the other. Thus Player I wins the log(n)+1 move game on G and H. His strategy is to play vertices v_i and v_j of G on the first two moves. As we have seen Player II is forced to answer with v_i and v_j from H, but now she has lost because the map between the first two elements is already not an isomorphism.

Thus two numbered graphs of size n are (log(n)+1)-equivalent only if they are identical. This is as expected because a pair of graphs G,H is indistinguishable to all log space Turing machines only if G=H.

Sometime after proving Theorem 7 we discovered to our surprise that with Suc we _can_ write a sentence of length O(log m) which says that G_m is in AGAP. This is done as follows: In a numbered graph a pair of vertices is endowed with an orientation. Thus a numbered copy of switch X is either right (orientation preserved) or wrong (orientation of the top pair is switched). Thus given a numbered graph which is either G_m or H_m we can tell which by adding up the number of wrong switches and seeing if it is odd or even.

To alleviate this problem we can replace the switch X in the above construction with a switch with m points. Thus to rember its orientation requires m bits rather than one. As above we can build graphs G_n' and H_n' which are $2^{\sqrt{\log(n)}}$-equivalent without successor. We conjecture that even with Suc they are indistinguishable.

<u>SECTION 4</u>: Extending Results to QN^S.

Proving lower bounds for $QN^S[f(n)]$ with $f(n) > \log(n)$ is much more subtle than for $QN[f(n)]$. We show below that quantifier rank lower bounds can no longer help us. By an <u>ordered</u> graph we mean a graph which comes with a valid successor relation. The following proposition shows that any property whatsoever of ordered graphs can be expressed in quantifier rank $\log(n)+3$.

<u>PROPOSITION 8</u>: Let C be any set of ordered graphs. Then for all n there exist sentences S_n of quantifier rank $\log(n)+3$ such that for all ordered graphs G of size $\leq$ n,

$$G \text{ is in } C \quad <-> \quad G \models S_n \quad .$$

<u>proof</u>: First we show that for any $i_0 \leq n$, we can write the formula $N_{i_0}(x)$, which means, "x is vertex number i_0 in the Suc ordering," in quantifier rank $\log(n)+1$. This is done by inductively defining the formulas $P_i(x,y)$ to mean that there is a successor path of length exactly i from x to y.

$$\begin{aligned}
P_1(x,y) &= Suc(x,y)\\
P_{2n-1}(x,y) &= (\exists z)(\ P_{n-1}(x,z)\ \&\ P_n(z,y)\ )\\
P_{2n}(x,y) &= (\exists z)(\ P_n(x,z)\ \ \&\ P_n(z,y)\ )
\end{aligned}$$

Now we identify the i^{th} point by saying that there is a path of length i from the first point to it:

$$N_i(x) \quad = \quad (\exists v_1)[\ P_{i-1}(v_1,x)\ \&\ (y)(\neg Suc(y,v_1))\ ]$$

$N_n(x)$ has quantifier rank $\log(n)+1$ and can also be written with $O[\log(n)]$ quantifiers using the abbreviation trick.

Now using $N_k(x)$ we can completely describe any graph G as follows:

$$F_G = \bigwedge_{i,j=1}^{n} (\exists x)(\exists y)[\ N_i(x)\ \&\ N_j(y)\ \&\ E^{i,j}(x,y)\ ]$$

Here $E^{i,j}(x,y) = E(x,y)$, or $\neg E(x,y)$ according as $E(v_i,v_j)$ holds or does not hold in G. Note that F_G has quantifier rank $\log(n)+3$. Let $C_n = \{G \mid G \in C\ \&\ |G| \leq n\}$. We define S_n as the disjunction over all G in C_n of F_G, i.e.,

$$S_n = \bigvee_{G \in C_n} F_G$$

This is the desired complete description of C_n. Although it may have length 2^{n^2}, S_n has quantifier rank only $\log(n)+3$. □

In spite of the above propostion there is still hope. Recall that from the last section we have a pair of structures G_n' and H_n' which are $2^{\sqrt{\log(n)}}$ equivalent but differ on the AGAP property. We conjecture that AGAP is not in $QN^S[2^{\sqrt{\log(n)}}]$.

Consider the set of all possible orderings of a graph G:

$$S(G) = \{\langle G, Suc_i \rangle \mid Suc_i \text{ is a successor relation on } G.\}$$

Thus $S(G_n')$ and $S(H_n')$ are families of ordered structures which we suspect cannot be separated by a sentence with $2^{\sqrt{\log(n)}}$ quantifiers. To make the notion "separated" precise we give the following

Definition: Let M and N be families of structures of the same finite type, T. We say that M and N are *k-inseparable*, $(M \sim_k N)$, if there is no sentence, F, from L(T) with k quantifiers such that:

$$M \models F \quad \text{and} \quad N \models \neg F,$$

i.e. every structure in M satisfies F and no structure in N does. Otherwise M and N are <u>k-separable</u>.

Clearly if we could show that $S(G_n')$ and $S(H_n')$ are $\log^k(n)$-inseparable it would follow that AGAP is not in $QN^S[\log^k(n)]$. The notion, "$AGAP_n$ and $\overline{AGAP_n}$ are $O[f(n)]$-separable," would be the same as the condition, "AGAP is in $QN^S[f(n)]$," if we had omitted the uniformity requirement in the definition of QN^S. Thus the following generalization of Theorem 2 holds:

<u>Proposition 9</u>: Let C be any set of ordered graphs. Then:

a. Suppose C is in $NSPACE^T[\log(n)]$ for some sparse oracle set T. Then C_n is $O[\log(n)]$-separable from $\overline{C_n}$, for every n.

b. Suppose C_n is $O[\log(n)]$-separable from $\overline{C_n}$, for every n. Then there is a sparse oracle, T, such that C is in $DSPACE^T[\log^2(n)]$.

<u>proof</u>: T is a <u>sparse</u> set if there are at most n^k objects in T of length n. The proof of this proposition is similar to that of Theorem 2. The differences are: In case (a), we must code into F_n the n^k elements of T that the log(n) space Turing machine can look at. In part (b), we must code the sentence F_n into $T \cap \{w \mid |w| = 2^{\log^2(n)}\}$. Note that any sentence with f(n) quantifiers and binary predicates is equivalent to some sentence of length $2^{f^2(n)}$. □

Proposition 9 is encouraging because it suggests that PTIME complete properties may be $O[\log(n))]$-inseparable from their complements. We close this section with a modified version of Ehrenfeucht games which test for separability:

<u>Definition</u>: Given families of structures, M and N, of the same finite type, we define the k-move <u>separability game</u> on M and N as follows:

On each of the k moves Player I chooses a point from <u>each</u> structure on one side or the other. Player II then chooses a corresponding point from each structure on the other side. II is allowed to make copies of structures so that she may choose several different answers from the same structure.

We say that Player II wins if there is a pair of structures and sequences of moves, $\langle G_i, m_1^i, \ldots, m_k^i \rangle$ and $\langle H_j, n_1^j, \ldots, n_k^j \rangle$ one from each side such that the map which sends constants from G_i to constants from H_j and maps m_r^i to n_r^j is an isomorphism of the induced substructures.

Theorem 10: Player II has a winning strategy for the k move game on M and N iff $M \sim_k N$.

proof: By induction on k.

k=0. Here if Player II wins then there is a pair of structures $G \in M$ and $H \in N$ whose constants are isomorphic. It follows that G and H satisfy all the same quantifier free formulas and so $M \sim_0 N$. Conversely if there is no such pair then the quantifier free formula, F_0, which is a disjunction of all the isomorphism types of constants from M is satisfied by all of M and none of N.

Inductively, assume that the r+1 quantifier formula $(\exists x)P(x)$ is true in M and false in N. Then Player I's first move will be to choose a point m_1^i from each $G_i \in M$ in such a way that $G_i \models P(m_1^i)$. No matter what II does, no structure $H_j \in N$ will satisfy $P(n_1^j)$. Think of the language as now having a new constant symbol c_1. Thus $\langle M, m_1 \rangle \models P(c_1)$ and $\langle N, n_1 \rangle \models \neg P(c_1)$ so by induction Player I wins.

Conversely assume that $M \sim_{r+1} N$ and let Player I choose m_1^i from each $G_i \in M$. Let $F_1, \ldots, F_s$ be a list of all the r quantifier formulas such that:

$$\langle M, m_1 \rangle \models F_i(c_1) \qquad i=1, \ldots, s$$

Therefore, $$M \models (\exists x)F_i(x) \qquad i=1, \ldots, s.$$

Thus for each i there is some G_i with $G_i \models (\exists x)F_i(x)$. Player II can play these s witnesses from the appropriate G_i's and forget about the rest of N. Note that this is where the making of copies is needed in case $G_i = G_j$ for $i < j < s$. Thus Player II can preserve the condition that $M \sim_r N$ and so by induction she will win. □

Because we have not yet learned the tricks of the separability game we prefer to leave it here as a jumping off point for further research. The game is studied in more detail in the author's thesis, [Imm80], and we are (perhaps overly) optimistic about its becoming a viable tool for ascertaining some of the lower bounds which are "well believed" but have so far in the history of this young discipline escaped proof.

SECTION 5: Conclusions.

We have shown that quantifier number is another measure of space complexity. Thus combinatorial techniques in the spirit of Ehrenfeucht games seem likely tools for demonstrating lower bounds for space.

That the difficulty of expressibility is closely tied to computational complexity is no accident. The deep connections between logic and complexity theory are inescapable. Just think, for example, of the link alternation (fundamentally an attribute of quantifiers, not Turing machines) gives in both directions between time and space. We believe that the notion of a property being expressible in some language is much simpler to understand and to prove things about than its being checkable by some Turing machine.

Finally, we expect further research in at least the following directions:

1: Characterize the difference between QN and QN^S. We know that for some "natural" problems like connectivity Suc(-,-) gives no gain in expressibility, whereas for other problems, such as counting the size of some set, there is an exponential gain. It would be very useful if there were some criterion to determine whether or not Suc(-,-) will help in a certain case.

2: R. Fagin and others have studied the notion of sentences probably holding in finite structures. It seems to me that an average successor relation would not help to separate the graphs G and H (mentioned above) which differed on a PTIME property. Thus there is hope of proving that the set of short sentences which hold for most successors is the same for G as for H. A similar idea would be to modify the notion of forcing in model theory (an adaption by Robinson of work of Cohen) to determine which short sentences are true for a "generic" successor.

3: NSPACE[log(n)] can be simulated by an existential sentence of quantifier rank log(n) and size O(n), or by a sentence with O(log(n)) alternating quantifiers. This mirrors the simulation of NSPACE by Parallel Time and by Alternating Time, respectively. In the first case the number of quantifiers corresponds to the number of processors in the parallel computation. This insight may lead to a new technique for analyzing parallelism and the time versus number of processor trade off.

4: The way we added Suc(-,-) is a bit strange. F_n expresses property P in QN^S if for all structures G of size n and for all binary relations Suc(-,-) such that Suc(-,-) is a valid successor relation on the universe of G, G is in P if and only if <G,Suc(-,-)> satisfies F_n. We can consider adding relations with other properties besides successor. One such property which we call a "marking", captures PTIME. It would be lovely to have a coherent theory of the increase in expressibility gained by adding an arbitrary relation with a given property.

5: An investigation of the classes $QN^S[f(n)]$, for $f(n) < \log(n)$, is needed. An intriguing fact is that a regular set such as EVEN requires log(n) quantifiers even with successor, and yet the set, CLIQUE(k), of graphs which contain a k-clique only needs a constant number of quantifiers. The latter class seems to require $TIME[n^k]$.

Acknowledgements: I would like to thank my advisor Juris Hartmanis for his kind help and excellent advice. Thanks also to John Hopcroft, Albert Meyer, and Michael Morley for helpful discussions.

REFERENCES

[Bor77]: Borodin,A., "On Relating Time and Space to Size and Depth," SIAM J. on Computing, Vol. 6, No. 4, Dec. 1977, pp. 733-744.

[ChSt76]: Chandra,A., Stockmeyer,L., "Alternation," Proc. 17th FOCS, 1976, pp. 98-108.

[Ehr61]: Ehrenfeucht,A., "An Application of Games to the Completeness Problem for Formalized Theories," Fund. Math, Vol. 49, 1961, pp. 129-141.

[End72]: Enderton,H.,A Mathematical Introduction to Logic,Academic Press, 1972.

[Fag74]: Fagin,R., "Generalized First-Order Spectra and Polynomial-Time Recognizable Sets," in: Complexity of Computation, (ed. R.Karp), SIAM-AMS Proc. 7, 1974, pp. 43-73.

[FiRa74]: Fischer,M., Rabin,M., "Super-Exponential Complexity of Presburger Arithmetic," in: Complexity of Computation, (ed. R.Karp), SIAM-AMS Proc. 7, 1974, pp. 27-41.

[Fra54]: Fraisse,R., "Sur les Classifications des Systems de Relations," Publications Sc. de l'Universite d'Alger, I, 1954.

[Gol77]: Goldschlager,L.,The Monotone and Planar Circuit Value Problems are Log Space Complete for P, SIGACT News, Vol. 9, No. 2, 1977.

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

[HPV77]: Hopcroft,J., Paul,W., Valiant,L., "On Time Versus Space," JACM, Vol. 24, No.2, 1977, pp. 332-337.

[Imm79]: Immerman,N., "Length of Predicate Calculus Formulas as a New Complexity Measure," Proc. 20th FOCS, 1979, pp. 337-347.

[Imm80]: __________, Ph.D Thesis, Cornell University, 1980.

[Jon75]: Jones,N., "Space-Bounded Reducibility Among Combinatorial Problems," JCS, 11, 1975, pp. 68-75.

[Koz76]: Kozen,D., "On Parallelism in Turing Machines," Proc. 17th FOCS, 1976, pp. 89-97.

[Sav70]: Savitch,W., "Relationships Between Nondeterministic and Deterministic Tape Complexities," J. Comp. System Sci. 4, 1970, pp. 177-192.

[Sav73]: Savitch,W., "Maze Recognizing Automata and Nondeterministic Tape Complexity," J. Comp. System Sci. 7, 1973, pp. 389-403.

[SaSt79]: Savitch,W., Stimson,M., "Time Bounded Random Access Machines with Parallel Processing," JACM Vol. 26, No. 1, 1979, pp.103-118.

[Sto77]: Stockmeyer,L., "The Polynomial-Time Hierarchy," Theoretical Comp. Sci. 3, 1977, pp. 1-22.