JOURNAL OF COMPUTER AND SYSTEM SCIENCES 22, 384–406 (1981)

Number of Quantifiers Is Better Than Number of Tape Cells*

NEIL IMMERMAN

Department of Mathematics,
Tufts University, Medford, Massachusetts 02155, and
Laboratory for Computer Science,
Massachusetts Institute of Technology,
Cambridge, Massachusetts 02139

Received January 5, 1981; revised January 15, 1981

We introduce a new complexity measure, $QN[f(n)]$, which measures the size of sentences from predicate calculus needed to express a given property. We show that:

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

Fraisse–Ehrenfeucht games are used to prove sharp lower bounds in the measure.

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 bona fide complexity measure which is not based on a machine model. 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 is nonetheless 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 lower bound techniques have nothing to do with the more usual methods of complexity theory involving complete sets or diagonalization.

We hope to convince the reader that it makes more intuitive sense 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 of Fagin (see [5]). He proved the following:

* Research partly supported by NSF Grant MCS 78-00418.

0022-0000/81/030384-23$02.00/0

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

1. $F = \exists P_1 \cdots \exists P_k \, \Phi(P_1 \cdots P_k)$, *where* $P_1 \cdots P_k$ *are predicate symbols and* Φ *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. It is difficult to show lower bounds for the expressibility of second order sentences. Instead we examine first order sentences which, we found, mimic computations much more closely. Considering, for example, graph problems, 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 $\mathrm{QN}[f(n)]$ 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. These sentences 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, $E(\text{-}, \text{-})$, representing the edge relation. This language seems sufficient 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 $\mathrm{QN}^S[f(n)]$ be the family of properties expressible with $f(n)$ quantifiers in a language that includes $\mathrm{Suc}(\text{-}, \text{-})$, a successor relation. We can now show that $\mathrm{NSPACE}[f(n)]$ is contained in $\mathrm{QN}^S[(f(n))^2/\log n]$.

We will say that C is in $\mathrm{QN}[f(n)]$ only if there is a *uniform* sequence of sentences expressing C. The uniformity (in the sense of Borodin, see [1]) allows us to prove $\mathrm{QN}[g(n)] \subseteq \mathrm{DSPACE}[g(n) \log n]$, and thus:

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

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 $\mathrm{QN}[f(n)]$ we mean that *no sentence with* $f(n)$ *quantifiers* expresses C for structures of size n.

The *quantifier rank* 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 *Ehrenfeucht–Fraisse game* 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 [3] and [8].

Ehrenfeucht–Fraisse games provide 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 $\mathrm{QN}[f(n)]$.

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 game argument. We show that without successor quantifier rank $\log^k(n)$ 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 $\bigcup_{k=1,2,\ldots} \mathrm{SPACE}[\log^k(n)]$.

Making the above result go through with successor is a major open problem. We show that quantifier rank is no longer the right thing to check. In a language with successor, any property whatsoever of graphs of size n can be expressed by a sentence with 2^{n^2} quantifiers but quantifier rank only $\log n$. To make matters worse, two ordered graphs G and H satisfy all of the same $3\log(n)$ quantifier sentences in the language with successor 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.

We conclude this paper by proposing a few possible techniques for adding successor to the above result and thus proving that $\bigcup \mathrm{QN}^s[\log^k(n)] \not\supset P$. The most hopeful one at present is a modification of Ehrenfeucht–Fraisse 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–Fraisse game and so we are by no means proficient at playing it. And yet we wanted to present, as a pont of departure for furture research, what may become a viable technique for proving lower bounds.

0. Review of Some Notions from Logic

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

A *similarity type*, $\tau = \langle c_1 \cdots c_k, P_1 \cdots P_n \rangle$, 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 $\tau_g = \langle E, s, d \rangle$, where V is the set of vertices of G, and E^G is G's edge relation.

If τ is any type then $L[\tau]$, the language of τ, is the set of all sentences built up from the symbols of τ using &, or $\neg$, $\rightarrow$, variables x, y,..., and the quantifiers $\exists$ and $\forall$.

A sentence, F, in $L[\tau]$ is given meaning by a structure, S, of type τ as follows: The symbols from τ 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 $A = \forall x[x = d \text{ or } \exists y E(x, y)]$. A is in $L[\tau_g]$. 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 quantifiers in F. Inductively:

$$qr[(\forall x)\, B] = qr[(\exists x)\, B] = qr[B] + 1,$$

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

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

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

1. The Quantifier Measure

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

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

$$G \text{ is in } C \leftrightarrow G \models F_n.$$

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

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

Thus C is in $\mathrm{QN}[h(n)]$ if there is a uniform sequence of sentences whose nth member has $O[h(n)]$ quantifiers and expresses the membership property of C for structures of size n. Our condition (c) is analogous to Borodin's notion of a problem's circuit depth in which he considers uniform sequences of boolean circuits (see [1]).

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 $\tau_g = \langle E(\cdot, \cdot), s, d\rangle$. Membership in GAP is known to be complete for $\mathrm{NSPACE}[\log n]$. (See [16].)

Theorem 1. GAP *is in* $\mathrm{QN}[\log n]$.

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.

$$P_1(x, y) = (x = y) \text{ or } E(x, y),$$

$$P_{2k}(x, y) = \exists z(P_k(x, z) \,\&\, P_k(z, y)).$$

The sentence P_n has quantifier rank $\log n$ and $n-1$ existential quantifiers. Using a familiar trick, (see [7] or [16]), we can add universal quantifiers and reduce the total number of quantifiers to 3 log (n):

$$A_1(x, y) = P_1(x, y),$$

$$A_{2k}(x, y) = \exists z \forall u \forall v((u = x \,\&\, v = z \text{ or } u = z \,\&\, v = y) \to A_k(u, v)).$$

Thus, leting $F_n = A_n(s, d)$, 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.

$$\text{EVEN} = \{G \mid |G| \text{ is even}\},$$

is not in QN$[\log n]$.

To allow the sentences to simulate Turing machines it suffices to give them 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 [16]. 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 τ is in QN$^s[h(n)]$ if there exists a sequence of sentences $F_1, F_2, \ldots$, from $L[\tau \cup \{\text{Suc}\}]$, and a constant k such that:

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

$$(G \in G) \leftrightarrow \langle G, \text{Suc}(-, -)\rangle \models F_n.$$

b. F_n has at most $k \cdot h(n)$ 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 $L[\tau \cup \{\text{Suc}\}]$ which give the same answer for any successor relation and express C.

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

THEOREM 2. *Let* $s(n) \geqslant \log(n)$. *Then*:

$$\text{NSPACE}[s(n)] \subseteq \text{QN}^s[s(n)^2/\log n] \subseteq \text{DSPACE}[s(n)^2].$$

Proof. We start with the second inclusion. We show that $\mathrm{QN}[g(n)]$ is contained in $\mathrm{DSPACE}[g(n)\log(n)]$. Given input structure G of size n, we can certainly generate F_n in the given space by the uniformity condition. Check the truth of a $g(n)$ quantifier sentence in $\mathrm{DSPACE}[g(n)\log(n)]$ as follows: Cycle through the sentence with all possible values of the quantified variables. If F_n is of the form $\exists x P(x)$ then we test the truth of $P(1),\ldots, P(n)$. Each variable requires $\log n$ bits and at most $g(n)$ 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.

The first inclusion: We will code a Turing machine instantaneous description (ID) of size $s(n)$ with $O[s(n)/\log(n)]$ variables. The idea is that each variable takes on a value from 1 to n and so may be thought of as $\log(n)$ bits. Details of this coding are given in Lemma 2a. An ID consists of a state, the location of the input head, and the $s(n)/\log(n)$ variable work tape. The read head requires a constant number of variables—for a graph two variables u, v give that element of the adjacency matrix being scanned. It is 1 or 0 according as $E(u, v)$ is true or false. One use of Suc is to efficiently code $\log(n)$ bits of worktape into one variable, but the essential use is to say that the read head has moved one space to the right.

We must now write $P_1(\mathrm{ID}_a, \mathrm{ID}_b)$ meaning that ID_b follows from ID_a in one step of M. The $s(n)$ bit work tape is coded with $s(n)/\log(n)$ pairs of vertices $\langle x_j, y_j\rangle$. All of the y_j's but one will be 0. The nonzero y_j will be vertex number 2^i when the work head is looking at bit i of x_j. Let $On(x, y)$ mean that $y = 2^i$ and bit i of x is on. We will see in Lemma 2a that $On(x, y)$ may be written with $O[\log n]$ quantifiers.

It follows that $P_1(\mathrm{ID}_a, \mathrm{ID}_b)$ may also be written with $O[\log n]$ quantifiers. We must say that some y_j is nonzero, the nonneighboring $\langle x, y\rangle$ pairs are unchanged, and $\langle x_j, y_j\rangle$ and its neighbors (in case the workhead happens to move to an adjacent block) are changed as per the rules of M. Note that to say, "Some y_j has property P", we write:

$$\exists y(y = y_1 \text{ or } \cdots \text{ or } y = y_{s(n)/\log(n)}) \,\&\, P(y)$$

Thus the "On" predicate need by written only once and P_1 may be written with $O[\log n]$ quantifiers as claimed. As in the proof that GAP is in $\mathrm{QN}[\log n]$ we can now assert that there is a computation path of length $c^{s(n)}$ using $O[s(n)]$ ID's. Thus the total size of P_n is $s(n)^2/\log(n)$ as required. We give the proof of Lemma 2a below, thus completing the proof of Theorem 2. ∎

LEMMA 2a. *$On(x, y)$ may be written in $QN^s[\log n]$.*

Proof. We build up to "On" with a sequence of inductive definitions, repeatedly using the abbreviation trick as in [7]. Thus each of the following may be written in $\mathrm{QN}^s[\log n]$ by using the previous one.

(a) $\mathrm{Plus}_n(x, y, z) = (x \leqslant n) \,\&\, (x + y = z)$.

(b) $Q_n(x_1 \cdots x_{\log n}) = (x_1 = 1) \,\&\, \bigwedge_{0<i<\log n}(x_{i+1} = x_i + x_i)$.

(c) $R_n(y_1 \cdots y_{\log n}) = \exists x_1 \cdots x_{\log n}(Q(x_1 \cdots x_{\log n}) \ \& \ \bigwedge_{0<i<\log n+1}(y_i = 0 \quad \text{or} \quad y_i = x_i))$.

(d) $S_n(z, y_1 \cdots y_{\log n}) = R_n(y_1 \cdots y_{\log n}) \ \& \ z = y_1 + \cdots + y_{\log n}$.

(e) $On(z, x) = \exists y_1 \cdots y_{\log n}(S_n(z, y_1 \cdots y_{\log n}) \ \& \ \bigvee_{0<i<\log n+1}(x = y_i \neq 0))$.

For example, Plus is defined as follows:

$$\text{Plus}_1(x, y, z) = [x = 0 \ \& \ y = z] \text{ or } [x = 1 \ \& \ \text{Suc}(y, z)];$$

$$\text{Plus}_{2k}(x, y, z) = \exists u\, v\, w(\text{Plus}_k(u, v, x) \ \& \ \text{Plus}_k(u, y, w) \ \& \ \text{Plus}_k(v, w, z)).$$

As before we can use the abbreviation trick to write Plus_k only once on the right. Note that we use the symbols 0 and 1 for convenience but they are of course definable from Suc.

Note also that to get each next equation one cannot simply use the previously defined formula. For example, to get Q_n a first try might be:

$$Q_n(x_1 \cdots x_{\log n}) = (x_1 = 1) \ \& \ \text{Plus}_n(x_1, x_1, x_2) \ \& \cdots \& \ \text{Plus}_n(x_{\log n-1}, x_{\log n-1}, x_{\log n}).$$

However this requires $\log^2(n)$ quantifiers. We must conglomerate Plus_n and Q_n into

$$C_n(x_1 \cdots x_{\log n}, u, v, w) = Q_n(x_1 \cdots x_{\log n}) \ \& \ \text{Plus}_n(u, v, w)$$

and define them simultaneously, using the abbreviation trick. ∎

2. Ehrenfeucht–Fraisse Games

In this section we will employ Ehrenfeucht–Fraisse games to obtain lower bounds for the quantifier measure. These games are due to Fraisse and Ehrenfeucht. (See [8] or [3] 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, τ, 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 *Player II wins* 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 τ. For example, if G and H are of type $\tau = \tau_g$, then $E(s, g_1)$ holds in G just if $E(s, h_1)$ holds in H.)

We say that two structures of type τ are *n-equivalent* if they satisfy all the same sentences in $L[\tau]$ of quantifier rank n. The fundamental fact about Ehrenfeucht–Fraisse games is:

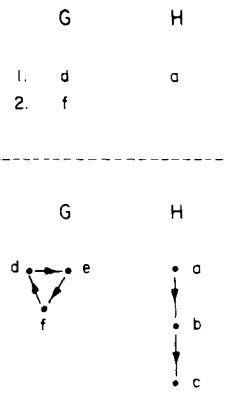

FIG. 1. An Ehrenfeucht–Fraisse game.

THEOREM 3 (Fraisse, Ehrenfeucht). *Player II has a winning strategy for then move game on A and B if and only if A is n-equivalent to B.*

As an example, consider the graphs G and H of Fig. 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, $S = \forall x \exists y E(y, x)$. 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.

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–Fraisse 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 Fig. 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.

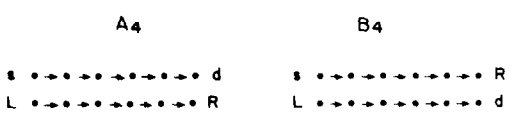

FIG. 2. A_m and B_m.

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 ith vertex in some row of A (or B), II will always answer with the ith 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), and at the same distance. If x is not within 2^k steps of such a point then II may answer with any point not within 2^k steps of an endpoint or chosen point.

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. ∎

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 GAP 1 and discussed in [10] and [14], is in DSPACE[log$n$].

As promised we now show that $L[\tau_g]$, 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 in order to count them.

PROPOSITION 5. EVEN, *the set of graphs with an even number of vertices, is in* DSPACE[log n], *(in fact it is in* DSPACE[0]), *but it is not in* QN[$h(n)$] *for any* $h(n)$ *asymptotically less than* n.

Proof. We already know by Theorem 2 that EVEN is in $\text{QN}^s[\log n]$. 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$[\log^k(n)]$. The graphs there are connected and of bounded degree. We feel that the latter example concerns itself with time versus space.

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 $x_1 \cdots x_n$. We say that b is *reachable* from a iff:

1. $a = b$;

or

2. a is marked "&," $n \geqslant 1$, 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 Fig. 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. Now define AGAP to be the set of alternating graphs inwhich d is reachable from s.

PROPOSITION 6. AGAP *is complete for polynomial time with respect to log-space reducibility.*

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 [2] or [15].) Boolean circuit value problems which are very similar have previously been shown to be complete for P. See for example [9]. ∎

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

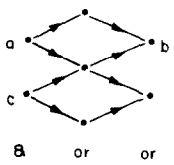

FIG. 3. An alternating graph.

THEOREM 7. *Let $f(n)$ be any function that is asymptotically less than $2^{(\log n)^{1/2}}$. Then* AGAP *is not in* QN$[f(n)]$. *In particular,* AGAP *is not in* QN$[\log^k(n)]$ *for any* k.

Proof. For all sufficiently large m, we produce graphs G_m and H_m with the following properties:

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

LEMMA 7a. *Let X be the alternating graph pictured in Fig. 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.*

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:

$$1 = [(e) \text{ or } (f)] = [(3 \,\&\, 5) \text{ or } (4 \,\&\, 6)];$$

$$2 = [(g) \text{ or } (h)] = [(3 \,\&\, 6) \text{ or } (4 \,\&\, 5)].$$

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.

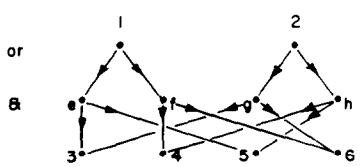

FIG. 4. Switch X.

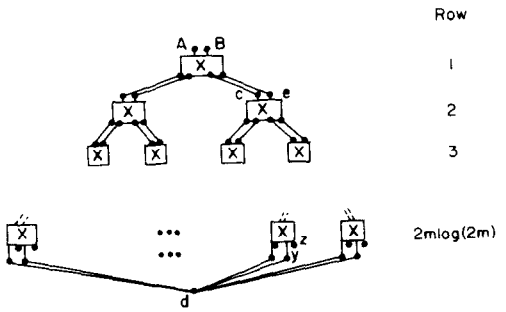

FIG. 5. $P_m(\text{if } s = A)$, $Q_m(\text{if } s = B)$.

The proof of Theorem 7 proceeds as follows: First we produce exponentially large graphs P_m and Q_m built up from switch X. P_m and Q_m differ on the AGAP property but we will see in Lemma 7b that they are m-equivalent. The final and most technical part of the proof is to reduce P_m and Q_m to G_m and H_m, graphs of size about $m^{\log m}$ which retain the above properties. Figure 5 shows $2^{1+2m\log(2m)} - 1$ copies of the switch X, arranged in a binary tree. Let P_m be the graph pictured in Fig. 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 $2^{2m\log(2m)}$ pairs on the bottom row. With only m moves, Player I cannot eliminate all of these possibilities.

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

We play a modified Ehrenfeucht–Fraisse game on $T_{2m\log(2m)}$, call it the "on–off" game. On each move of his 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 "self-consistent" we mean that with the labelling of row r removed, the labelling of the $2^k - 1$ points may be extended to a consistent labelling of the entire tree. The proof is 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 descendents 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 from Lemma 7c that Player II wins the $2m$-move on–off game on $T_{2m\log(2m)}$. We state this as a lemma so we may reuse the proof later on:

LEMMA 7d. *Suppose that an appropriate graph G with $k \cdot 2^k$ rows satisfies Lemma 7c. Then Player II wins the 2^k move on–off game on G.*

Proof. Player II's strategy is to answer "off" whenever this is consistent with the previously labelled points and with the top point being on. We show that no point in row $r \cdot k$ must be labelled on until the rth move.

Assume that Player II has successfully followed her strategy for the first $r - 1$ moves and that on move r Player I chooses point p in row $r \cdot k + 1$. We must show that II may answer "off." Since there are $r \cdot k$ rows above p, and only $r - 1$ previously chosen points, there must be a block, B, of $k - 1$ consecutive rows with no chosen points. Consider the first point chosen by Player I which lies below B. By Lemma 7c that point could have been consistently chosen off. Therefore Player II's strategy was to label it off. Similarly Player II must have answered "off" to each additional point below B. In particular, p is below B and may be labelled "off."

It follows that in the 2^k move game Player II's strategy allows her to label the chosen points consistently and not label any of the points on the bottom row "on." Thus she wins. ∎

With $k = \log(2m)$, we see that Player II wins the $2m$-move on-off game on $T_{2m\log(2m)}$. We can now play the original m-move Ehrenfeucht–Fraisse game as follows: (See Fig. 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 e, 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. If either of these descendants is "on" then the moves induce one of the automorphisms of switch X listed in Lemma 7a. Player II should perform this automorphism on the switch in question and answer accordingly.

We claim that this is a winning strategy for Player II; i.e., there is an isomorphism between the chosen points from the two graphs. The rule that in the on–off game Player II must call the top point on and the bottom points off assures that s will be answered by s and any point touching d will be answered by a point also touching d. The fact that Player II wins the on–off game indicates that any triple of neighboring switches is matched up correctly. 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 about $m^{\log m}$ vertices above row m but still has the property that no point in block k can be forced on before the kth move. We define D_k below algebraically, but please refer to Figs. 6 and 7 and 8 which show portions of D_1, D_2, and D_3, respectively.

$$\text{Vertices}(D_k) = \{\langle x_1 \cdots x_k, r\rangle \mid r = b \cdot k + p, p < k,$$
$$0 \leqslant x_i \leqslant b+1 \quad \text{for} \quad 1 \leqslant i \leqslant p \;\&\; 0 \leqslant x_i \leqslant b \quad \text{for} \quad p < i \leqslant k\}.$$

$$\text{Edges}(D_k) = \{(\langle x_1 \cdots x_k, r\rangle, \langle x_1 \cdots x_k, r+1\rangle) \mid r \geqslant 0\}$$
$$\cup \{(\langle x_1 \cdots x_k, r\rangle, \langle x_1 \cdots x_p, x_{p+1}+1, \ldots, x_k, r+1\rangle) \mid r \equiv p \bmod(k)\}.$$

FIG. 6. Four blocks of D_1.

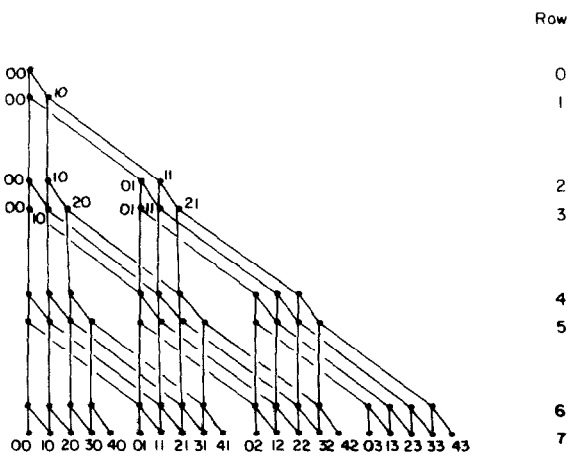

FIG. 7. Four blocks of D_2.

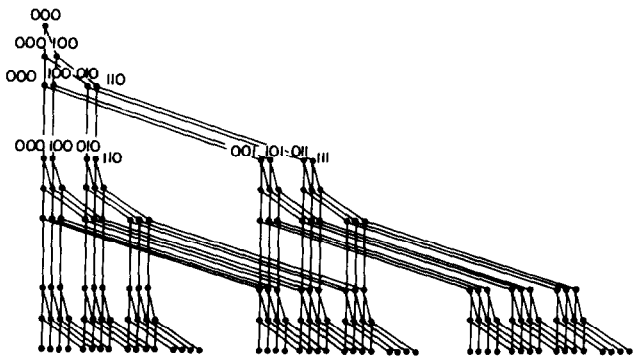

FIG. 8. Three blocks of D_3.

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 7e. *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 consistent with row r.*

Proof. We break the proof into parts. First assume that the $2^k - 1$ chosen points lie *on* row $r + k$. The proof is 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. Clearly one of these marks the chosen point correctly.

Let the jth *block* of D_k be the k consecutive rows numbered $j \cdot k$ to $j \cdot k + k - 1$. Note that in passing from the ith row of one block to the ith row of the next block the range of each coordinate is stretched by 1. Assume for convenience that row r is at the bottom of block j.

Let the ith *column* of D_k be those points with kth coordinate equal to $\min(i, m_r)$, where m_r is the maximum possible kth coordinate at the given row. Note that the ith column of D_k is a copy of D_{k-1} with every kth row repeated. In particular the ith column of the jth block of D_k is a copy of the jth block plus the first row of the $j + 1$st block of D_{k-1}. (Note that the way we have drawn Figs. 7 and 8 columns correspond to vertical sections of the graph. Suppose row r were not at the bottom, but rather at the sth row of some block. We would then redefine the notion of block so that r is the bottom row of a new block. Columns would be the points agreeing on the sth coordinate rather than the kth.)

Inductively consider any labelling of row r of D_k together with a choice of $2^k - 1$ labelled points on row $r + k$. Clearly at most one of the columns of row $r + k$ has 2^{k-1} chosen points. Let this i_0th column of row $r + 1$ be chosen in any consistent

fashion. All the other columns have at most $2^{k-1}-1$ chosen points in row $r+k$. Thus by induction all the other columns of row $r+1$ may be chosen as we please. Thus we can counteract the i_0th column and choose row r as we please: Choose column i_0-1, row $r+1$ to be the sum of the desired column i_0-1, row r, and column i_0, row $r+1$. Next choose columns i_0+1, i_0-2, and so on.

To complete the full lemma we must generalize our inductive assumption:

CLAIM. *Suppose that row r of D_k is entirely labelled. Further assume that some of the edges are marked "¬" meaning that the signal going through that edge is reversed. 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 entire graph giving the desired row r.*

Proof. As above at most one column of D_k say i_0 has at least 2^{k-1} chosen points. Consider the wedge, W below the i_0th column of row $r+1$. W consists of column i_0 block $j+1$, columns i_0, i_0+1 block $j+2$, and so on. See Fig. 9 for a schematic view of W.

Label the points of W in any fashion consistent with the labelling of the 2^k-1 chosen points, taking into account the edges marked "¬." Now consider column i_0-1. For each edge e from a to b in column i_0-1 it may be the case that there is also an edge from c to b where c is a point in W labelled "on." If so we mark e "¬" because the value of b will be the opposite of the value of a. We next merge the repeated pair of rows in each block of column i_0-1, thus making a true copy of D_{k-1}. Suppose d lies above b which lies above a, with edges f from b to d, and e from a to b. The merging involves deleting b and replacing e and f by f', a new edge from a to d. The label of f' will be ¬ just if exactly one of e and f was labelled ¬. Note that the labelling is still self-consistent because W was labelled in a consistent way. By our inductive assumption row $r+1$ of column i_0-1 may be filled in as we please.

Labelling column i_0-1 induced ¬'ed edges in column i_0-2. How can we insure that this introduces no inconsistencies? The answer is as follows: Consider any

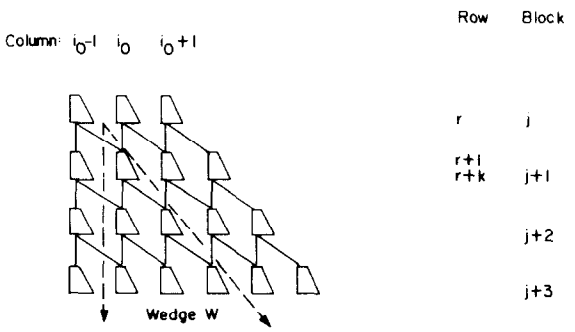

FIG. 9. Schematic view of wedge W.

consistent labelling, L, of columns 1 to $i_0 - 1$. For each of the originally labelled points, u, in columns 1 to $i_0 - 2$ let u' be the corresponding point in column $i_0 - 1$. Note that any change in the labelling of column $i_0 - 1$ either fixes or switches both u and u'. Add $\langle u', L(u')\rangle$ to the list of labelled points. Now by induction find a labelling of column $i_0 - 1$ consistent with the constraints and generating the desired section of row $r + 1$. This labelling does not change L's value for the originally labelled points in columns 1 to $i_0 - 2$. Thus the $\neg$'s induced by column $i_0 - 1$ remain consistent.

It follows that columns $i_0 - 2$, $i_0 - 3$, and so on, may be filled in as we please. Similarly we can use the diagonals to the right of W to fill in columns $i_0 + 1$, $i_0 + 2$,..., of row $r + 1$. Thus we can generate row r as we please. ∎

It follows from Lemmas 7d and 7e that Player II wins the 2^k move on–off game on the first 2^k blocks of D_k. Her strategy is to say "off" until a point is forced on. No point p can be forced on unless more than $2^k - 1$ points have been chosen or there is a point marked on within k rows of p. Thus 2^k moves do not suffice to force on a point in the bottom row.

Let G_m and H_m be the graphs arising from the first $2m$ blocks of $D_{\log(2m)}$ by replacing vertices by the switch X, just as P_m and Q_m arose from the binary tree of height 2m $\log(2m)$. 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 and H_m is not in AGAP.

Our above remarks imply that Player II wins the $2m$ move on–off game on $D_{\log(2m)}$. Thus, as in the proof of Lemma 7b, 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 indices 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. 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 probably can write a sentence of length $O[\log n]$ which distinguishes G_m from H_m. 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 P_m or Q_m we can tell which by adding up the number of wrong switches and seeing if it is odd or even.

This does not quite work for distinguishing G_m from H_m because some of the switches in D_k have no effect—that is their signals lead to the top an even number of times. (See, for example, vertex 10 in row 3 of Fig. 6.) We believe (although we have not written out the details) that the pattern of which vertices count is simple enough to admit expression via a $\log(n)$ quantifier formula $C(x)$. If so then G_m can be distinguished from H_m with $\log(n)$ quantifiers by adding up the number of wrong switches y such that $C(y)$.

To alleviate this problem we can replace the switch X in the above construction with a switch with m points. Thus to remember its orientation requires m bits rather than one. As above we can build graphs G'_n and H'_n which are $2^{(\log n)^{1/2}}$ equivalent without successor. We conjecture that even with Suc they are indistinguishable.

4. 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 *ordered* 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 $\log(n) + 3$.

Proposition 8. *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 $\leqslant n$,*

$$G \in C \rightleftarrows G \models S_n.$$

Proof. First we show that for any $i_0 \leqslant n$, 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.

$$P_1(x, y) = \mathrm{Suc}(x, y);$$

$$P_{2n-1}(x, y) = \exists z[P_{n-1}(x, z) \,\&\, P_n(z, y)];$$

$$P_{2n}(x, y) = \exists z[P_n(x, z) \,\&\, P_n(z, y)\}.$$

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

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

$N_n(x)$ 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:

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

Here $E^{ij}(x, y)$ is $E(x, y)$, 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 $C_n = \{G \mid G \in C \,\&\, |G| \leqslant n\}$. 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 proposition there is still hope. Recall that from the last section we have a pair of structures G'_n and H'_n which are $2^{(\log n)^{1/2}}$ equivalent but differ on the AGAP property. We conjecture that AGAP is not in $\mathrm{QN}^s[2^{(\log n)^{1/2}}]$.

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

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

Thus $S(G'_n)$ and $S(H'_n)$ are families of ordered structures which we suspect cannot be separated by a sentence with $2^{(\log n)^{1/2}}$ 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, τ. We say that M and N are *k-inseparable* if there is no sentence, F, from $L[\tau]$ with k quantifiers such that:

$$M \models F \qquad \text{and} \qquad N \models \neg F;$$

i.e., every structure in M satisfies F and no structure in N does. Otherwise M and N are *k-separable.*

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 $\mathrm{QN}^s[\log^k(n)]$. The notion, "AGAP_n and $\neg\mathrm{AGAP}_n$ are $O[f(n)]$-separable," would be the same as the condition, "AGAP is in $\mathrm{QN}^s[f(n)]$," if we had omitted the uniformity requirement in the definition of QN^s. Thus the following generalization of Theorem 2 holds:

PROPOSITION 9. *Let C be any set of ordered graphs. Then:*

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

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

Proof. T is a *sparse* set if there are at most n^k objects in T of length n. A SPACE$^T[f(n)]$ machine has a size $f(n)$ query tape on which it may write words and ask if they are in T. The proof of this proposition is similar to that of Theorem 2. The differences are:

For part (a), we must code into F_n the n^k elements of T that the log n space Turing machine can look at. Thus the formula, $P_1(\mathrm{ID}_a, \mathrm{ID}_b)$, saying that ID_b follows from ID_a in one step of M^T must include the disjunction over n^k possible questions to the oracle. However, all quantifiers may be placed outside this disjunction so the quantifier number is unchanged.

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 $c^{f^2(n)}$. We use the first $c^{f^2(n)}$ words of lenth $2^{\log^2(n)}$ to code F_n as a binary string. Thus there are at most $c^{f^2(n)}$ members of T of length $2^{\log^2(n)}$. Thus T is sparse. ∎

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–Fraisse games which test for separability:

DEFINITION. Given families of structures, M and N, of the same finite type, we define the k-move *separability game* on M and N as follows:

On each of the k moves Player I chooses a point from *each* 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, $\langle G_i, m_1^i \cdots m_k^i \rangle$ and $\langle H_j, n_1^j \cdots n_k^j \rangle$ 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 and N are k-inseparable.*

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 and N are 0-inseparable. 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. Them 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 and N are $r+1$-inseparable and let Player I choose m_1^i from each $G_i \in M$. Let $F_1 \cdots F_s$ be a list of all the r quantifier formulas with the new symbol c_1 that are true for each structure in M, that is:

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

Therefore,

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

Since $\exists F_i(x)$ cannot separate M from N there must be some H_i in N such $H_i \models \exists x F_i(x)$. Thus Player II can play these s witnesses from the appropriate H_i's and forget about the rest of N. Note that this is where the making of copies is needed in case $H_i = H_j$ for some $i \neq j$. Thus Player II can preserve the condition that M and N are r-inseparable and so by induction she will win. ∎

Little is known about how to play the separability game. We leave it here as a jumping off point for further research. We urge others to study it, hoping that the separability game may become a viable tool for ascertaining some of the lower bounds which are "well believed" but have so far escaped proof.

5. Conclusions

We have shown that quantifier number is another measure of space complexity. Thus combinatorial techniques in the spirit of Ehrenfeucht–Fraisse 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. (See [6].) It seems to me that an average successor relation would not help to separate the graphs G'_n and H'_n (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 of $O[\log n]$ 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 $\langle G, \mathrm{Suc}\rangle$ satisfies F_n. We can consider adding relations with other properties besides successor. One such property which we call a "marking," capures 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 $\mathrm{QN}^s[f(n)]$, 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 DTIME[n^k].

ACKNOWLEDGMENTS

I would like to thank my advisor Juris Hartmanis for his kind help and excellent advice. Thanks to John Hopcroft, Albert Meyer, and Michael Morley for very helpful discussions. Thanks also to the referees who pointed out errors and lack of clarity in some of the proofs.

REFERENCES

1. A. BORODIN, On relating time and space to size and depth, *SIAM J. Comput.* **6**, No. 4 (1977), 733–744.
2. S. CHANDRA AND L. STOCKMEYER, Alternation, *in* "Proceedings, 17th FOCS, 1976," pp. 98–108.
3. A. EHRENFEUCHT, An application of games to the completeness problem for formalized theories, *Fund. Math.* **49** (1961), 129–141.
4. H. ENDERTON, "A Mathematical Introduction to Logic," Academic Press, New York, 1972.
5. R. FAGIN, Generalized first-order spectra and polynomial-time recognizable sets, *in* "Complexity of Computation" (R. Karp, Ed.), SIAM-AMS Proc. 7, pp. 43–73, 1974.
6. R. FAGIN, Probabilities on finite models, *J. Symbolic Logic* **41**, No. 1 (1976), pp. 50–58.
7. M. FISCHER AND M. RABIN, Super-exponential complexity of presburger arithmetic, *in* "Complexity of Computation" (R. Karp, Ed.), SIAM-AMS Proc. 7, pp, 27–41, 1974.
8. R. FRAISSE, Sur les classifications des sytèmes de relations, *Publ. Sci. Univ. Alger* **1**, 1954.
9. L. GOLDSCHLAGER, The monotone and planar circuit value problems are log space complete for P. *SIGACT News* **9**, No. 2 (1977).
10. J. HARTMANIS, N. IMMERMAN, AND S. MABANEY, One-way log tape reductions, *in* "Proceedings, 19th FOCS, 1978," pp. 65–72.

11. J. HOPCROFT, W. PAUL, AND L. VALIANT, On time space, *J. Assoc. Comput. Mach.* **24**, No. 2 (1977), 332–337.
12. N. IMMERMAN, Length of predicate calculus formulas as a new complexity measure, *in* "Proceedings, 20th FOCS, 1979," pp. 337–347.
13. N. IMMERMAN, Ph. D. Thesis, Cornell University, 1980.
14. N. JONES, Space-bounded reducibility among combinatorial problems, . *Comput. Sci.* **11** (1975), 68–75.
15. D. KOZEN, On parallelism in turing machines, *in* "Proceedings, 17th FOCS, 1976," pp. 89–97.
16. W. SAVITCH, Maze recognizing automata and nondeterministic tape complexity, *J. Comput. System Sci.* **7** (1973), 389–403.
17. W. SAVITCH AND M. STIMSON, Time bounded random access machines with parallel processing, *J. Assoc. Comput. Mach.* **26**, No. 1 (1979), 103–118.
18. L. STOCKMEYER, The polynomial-time hierarchy, *Theoret. Comp. Sci.* **3** (1977), 1–22.