LENGTH OF PREDICATE CALCULUS FORMULAS AS A NEW COMPLEXITY MEASURE *

Neil Immerman

Department of Computer Science
Cornell University
Ithaca, N.Y. 14853

Abstract

We introduce a new complexity measure, QR[f(n)], which clocks the size of formulas 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 seperating Time and Space complexity classes because we show that:

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

Introduction and Summary:

For the purpose of analyzing the time and space requirements of computations, we introduce a new complexity measure. Rather than asking how many steps or how much tape is needed to accept a certain set of graphs, we examine the size of a formula in predicate calculus needed to describe the graph property in question.

The result is Quantifier Rank (QR) a bonafide complexity measure which is not based on a machine model. This provides new techniques and insights. In particular there are well established methods in logic to decide what can and cannot be said in various languages. These techniques enable us to prove sharp lower bounds having nothing to do with complete sets or diagonalization.

It turns out that QR 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 on the time versus space problem.

*Research supported by NSF Grant MCS 78-00418.

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

THEOREM (Fagin): A set, S, of structures is in NP if and only if there exists a formula, 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 formula.
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 formula. (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 formulas. Instead we examine first order formulas which, we found, mimic computations much more closely. Considering graph problems, for example, the length of the shortest formula 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 formulas we introduce the complexity measure QR[T,f(n)] which will be defined in Section 1. Informally, a set, S, of structures of type T is in QR[T,f(n))] if membership in S for those structures of size n is expressible by a formula of "size" f(n). By size we mean quantifier rank, the depth of nesting of quantifiers, (defined precisely in Section 0).

As suggested above QR is closely related to space. The intuitive similarity is that the quantifier rank of a formula, F, is the maximum number of variables which F can simultaneously consider. In Section 1 we prove that $QR[f(n)^2/\log n]$ contains NSPACE[f(n)] and is contained in $DSPACE[f(n)^2]$, thus making the relationship precise.

In Section 2 we consider a two person game with which we prove lower bounds for the quantifier measure. 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 formulas of quantifier rank n. The original treatment of these games appears in [Ehr61] and [Fra54].

Thus we have a technique for showing lower bounds. For example we prove that while, for a graph of size n, quantifier rank log(n) suffices to express the property, "There is a path from point s to point d," quantifier rank log(n)-2 is insufficient!

In Section 3 we present a more sophisticated game argument. We show that in a reduced language quantifier rank $(\log n)^k$ is insufficient to describe a set recognizable in polynomial time. If our proof went through for the full language we would have shown that P is not contained in $\bigcup_{k=1,2,\ldots} \mathrm{SPACE}[(\log n)^k]$.

In the final section we make some concluding remarks concerning QR and its relationship to the time versus space problem. We feel that quantifier rank and the associated Ehrenfeucht games are an interesting new tool for studying space complexity.

<u>SECTION 0</u>: Review of some notions from logic.

A <u>structure</u>, $S=\langle U,c_1^U,\ldots,c_k^U,P_1^U,\ldots,P_n^U\rangle$, 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 <u>similarity type</u>, $T=\langle c_1,\ldots,c_k,P_1,\ldots,P_n\rangle$, 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 formulas built up from the symbols of T using &, or, $\neg$, ->, =, variables x,y,z,...; and the quantifiers $(\exists x)$ and (x).

A formula, 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 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 <u>quantifier rank</u> of formula 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 QR[T,h(n)] if there exists a sequence of formulas $\{F_i | i=1,2,...\}$ from L(T) such that:

(i): For all structures, G, of type T, if $|G| = n$, then G is in C if and only if G satisfies F_n.
(ii): $qr[F_n] \leq h(n)$
(iii): The map $f:n \to F_n$ is "easy to generate", i.e. computable by a DSPACE[h(n)] Turing machine. Thus F_n is of length $<c^{h(n)}$.

Thus C is in QR[T,h(n)] if there are formulas of quantifier rank h(n) which express the membership property of C for structures of size n. Our definition 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. Membership in GAP is known to be complete for NSPACE[log(n)]. (See [Sav73].)

Theorem 1: GAP is in QR[T_g,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 formulas $P_i(a,b)$. $P_i(a,b)$ says that there is a path of length at most 2^i from a to b.

$$P_0(a,b) = (a=b) \text{ or } E(a,b)$$
$$P_{k+1}(a,b) = (\exists x)(P_k(a,x)\ \&\ P_k(x,b))$$

Putting $F_n=P_{\log(n)}(s,d)$, we see that F_n expresses the GAP problem for graphs of size n. Furthermore, F_n has quantifier rank log(n) and is generable in SPACE[log(n)]. #

Note: The formula F_n has 2^n existential quantifiers. Using a familiar trick, (see [FiRa74] or [Sav70]), we can add universal quantifiers and reduce the length of F_n to O(log(n)):

$$M_0(a,b) = P_0(a,b)$$
$$M_{k+1}(a,b) = (\exists z)(x)(y)[(x=a\ \&\ y=z) \text{ or } (x=z\ \&\ y=b)] \to M_k(x,y)$$

We will write, e.g., SPACE[T,f(n)] to denote the set of structures of type T accepted in SPACE[f(n)]. Although the complete problem GAP is in QR[T_g,log(n)], not all graph problems in NSPACE[T_g,log(n)] are also in QR[T_g,O(log(n))]. In Section 2 a counterexample is provided.

To allow them to simulate Turing machines it suffices to give the formulas access to the numbering of the vertices which the machines already have. Thus for each type T, we consider the type T^S of T together with the 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 formulas do not need Suc to express "natural" graph problems.

Note that any sequence of 0's and 1's may be thought of as the adjacency matrix of a graph, with a certain number of ommitted trailing 0's so that it is of size n^2. Thus we may identify SPACE[f(n)] with SPACE[T_g,f(n)]. In keeping with this we will write QR[f(n)] to mean QR[T_g^S,f(n)], the collection of properties describable in quantifier rank f(n) in the language of numbered graphs. The following theorem shows that QR is closely related to space.

Theorem 2: Let f(n) be any function such that f(n) $\geq$ log(n), and let T be any type. Then:

(a): Each problem in NSPACE[T,f(n)] is contained also in QR[T^S,$\frac{c(f(n))^2}{\log(n)}$] for some constant c.

(b): For any function g(n), QR[T,g(n)] is contained in DSPACE[g(n)log(n)].

Thus:

$$\text{NSPACE}[f(n)] \subseteq \bigcup_{k=1}^{\infty} \text{QR}\left[\frac{k(f(n))^2}{\log(n)}\right] \subseteq \text{DSPACE}[f(n)^2] .$$

proof: We sketch the proof of (a). 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)) varibles. (A similar technique appears in [Sto77].) It is not hard to prove by induction that log(n) quantifier rank suffices to say, "Digit i of element x is 0." Thus in quantifier rank log(n) we can say "ID_1 follows from ID_2 in one step." The length of a computation may be $c^{f(n)}$ so we need O[f(n)] id's to state that such a path exists.

(b): Given G of size n we can generate F_n with SPACE[g(n)]. Check the truth of a formula of rank g(n) in DSPACE[g(n)log(n)] as follows: Cycle through each branch of the formula with all possible values of the quantified variables. Each variable requires log(n) bits and g(n) of them must be remembered at once. #

SECTION 2: 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 appear 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, 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. (That is f preserves all of the symbols of T. For example, if T=T_g, E(s,g_1) holds in G just if E(s,h_1) holds in H.)

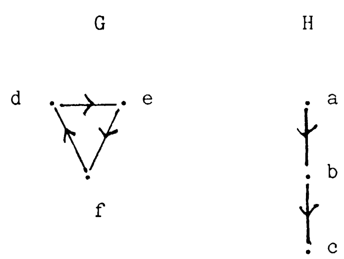
We say that two structures of type T are n-equivalent if they satisfy all the sames formulas in L(T) of quantifier rank n. The fundamental fact about Ehrenfeucht games is:

Theorem 3(Fraisse,Ehrenfeucht): 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	_

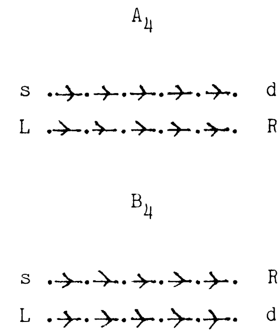


The next theorem uses Ehrenfeucht games to show that Theorem 1 cannot be improved except for the additive constant!

Theorem 4: GAP is not in $QR[T_g, \log(n)-2]$.

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:



We will now show that A_m is $(\log(n)-2)$-equivalent to B_m. From this it follows that no formula of quantifier rank $\log(n)-2$ can express the property, "There is a path from x 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 stepts 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. #

Theorem 4 goes through for T_g^S as well. 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 log(n)-2 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 disconnected graph. (The same example could be built with connected graphs of unbounded degree.)

Proposition 5: E3, the set of graphs with one third as many edges as vertices, is in DSPACE[log n], but is not in $QR[T_g, n/3]$.

proof: Clearly E3 is in DSPACE[log(n)]. Fix m and let n=3m. Define the graphs C_n and D_n as follows:

$$\text{Vertices}(C_n) = \{x_1,\ldots,x_m; y_1,\ldots,y_m; z_1,\ldots,z_m\}$$
$$\text{Edges}(C_n) = \{(x_i,y_i) \mid i=1,2,\ldots,m\}$$

$$\text{Vertices}(D_n) = \{b_1,\ldots,b_{m-1}; c_1,\ldots,c_{m-1}; d_1,\ldots,d_{m+2}\}$$
$$\text{Edges}(D_n) = \{(b_i,c_i) \mid i=1,2,\ldots,m-1\}$$

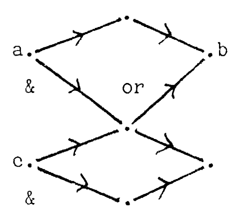
Thus C_n is in E3, but D_n is not. An Ehrenfeucht game shows, however, that C_n is n/3-equivalent to D_n. Player II's strategy is to match any b_i, c_i, d_i, with a x_j, y_j, z_j, respectively. Consistency must be preserved, so if x_2 is matched with b_4, then y_2 must be matched with c_4. Thus Player II wins, so E3 is not in $QR[T_g, n/3]$. #

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

SECTION 3: P-TIME and the QR 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 "&" 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.) Now define AGAP to be the set of alternating graphs inwhich d is reachable from s.

FIGURE 3: An Alternating Graph



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].) #

We must now add the predicate A(x) meaning that vertex x is marked "&". Let $T_{ag} = \langle E,A,s,d \rangle$, 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 for $L(T_{ag}^S)$ then we would have shown that P is not contained in SPACE[$(\log n)^k$].

<u>Theorem 7</u>: AGAP is not in QR[T_{ag},$(\log n)^k$] for any k.

<u>proof</u>: Fix m so large that $[2(\log m)]^{2k} < m$. We produce graphs G_m and H_m with the following properties:

(i): G_m and H_m both have fewer than $m^{2(\log m)}$ vertices. Thus $[\log |G_m|]^k < m$.
(ii): G_m is m-equivalent to H_m.
(iii): G_m is in AGAP, but H_m is not.

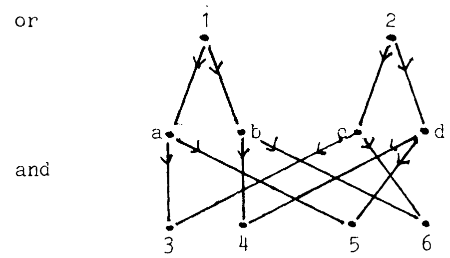
When (i),(ii), and (iii) are met we will have shown that in $L(T_{ag})$ quantifier rank $(\log n)^k$ does not suffice to express the Alternating Graph Accessibility Problem.

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:

(i): f switches 3 & 4 and 1 & 2, leaving 5 & 6 fixed.
(ii): g switches 1 & 2 and 5 & 6, leaving 3 & 4 fixed.
(iii): h switches 3 & 4 and 5 & 6, leaving 1 & 2 fixed.

<u>FIGURE 4</u>: Switch X



<u>proof</u>: 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:

$$1 = a \text{ or } b = (3 \& 5) \text{ or } (4 \& 6)$$
$$2 = c \text{ or } d = (3 \& 6) \text{ or } (4 \& 5)$$

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.

<u>FIGURE 5</u>: P_m (if s=A), Q_m (if s=B)

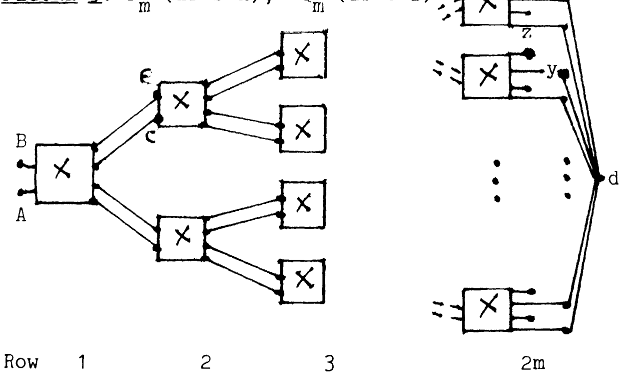


Figure 5 shows $2^{2m+1}-1$ copies of the switch X, arranged in a binary tree. Let P_m be the graph pictured in Figure 5, with s=A. Let Q_m be the same graph, but with s=B. Thus P_m is in AGAP while Q_m is not. However,

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.

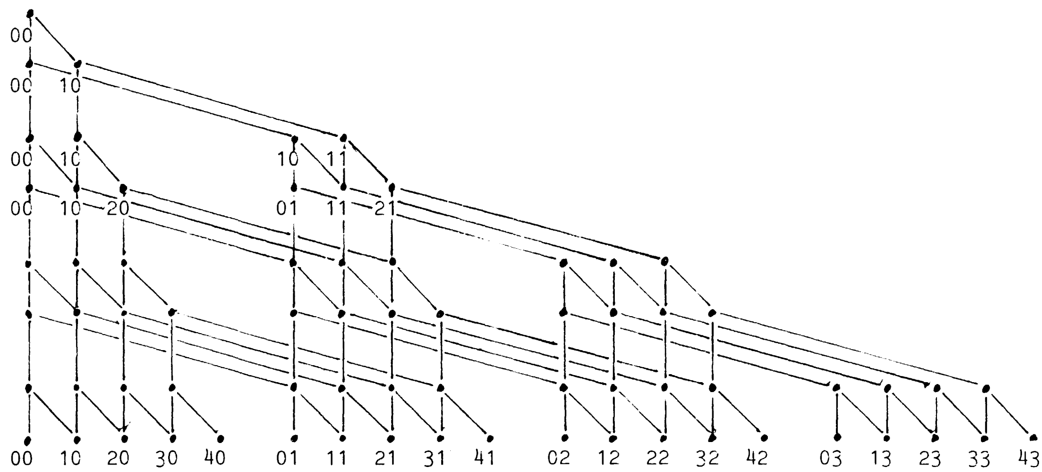
Inductively suppose that 2^k-1 points are labelled on or below row r+k. Let L be the set of left offspring in row r+1, R the set of right offspring. Clearly at most one of these sets, say L, has more than $2^{k-1}-1$ of its descendants labelled. Label all of the vertices in L in any consistent fashion. 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 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, and move accordingly. This proves Lemma 7b. #

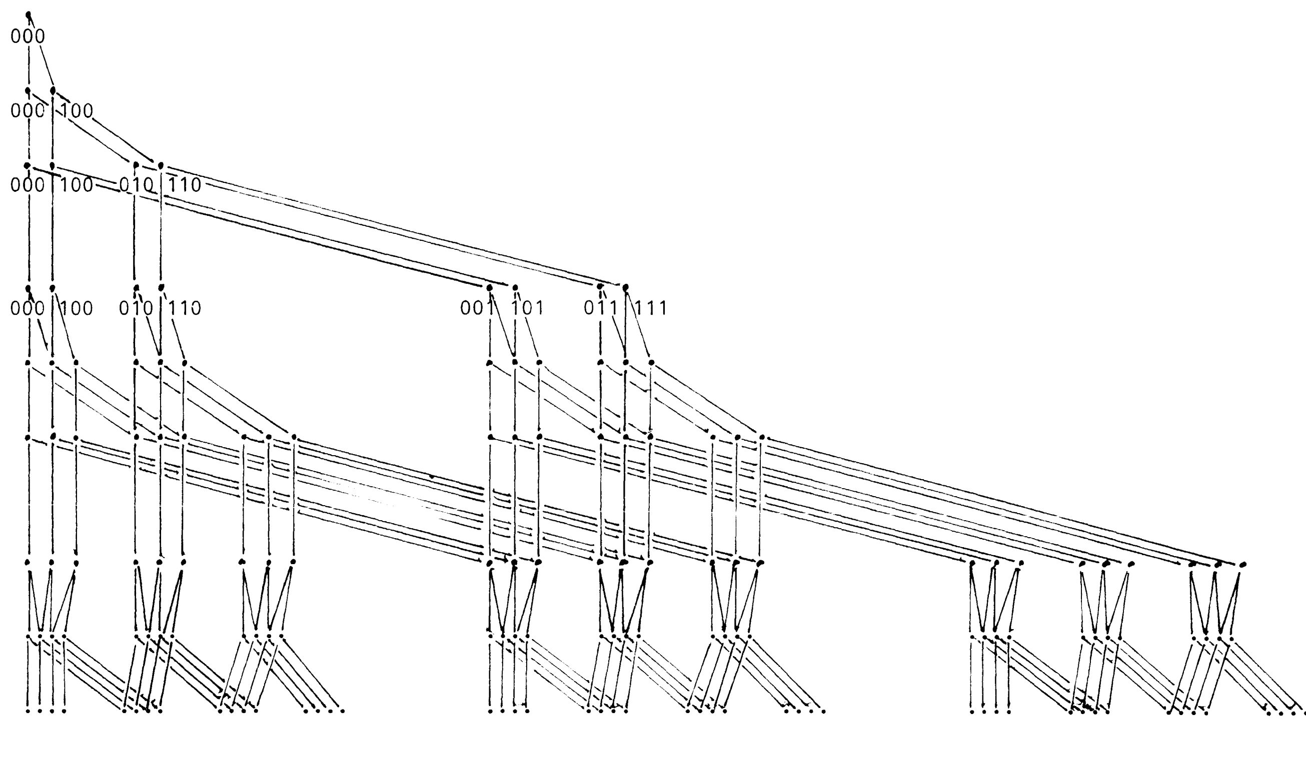
The final step of the proof is to introduce the graph $D_{\log m}$ to replace the binary tree in the above construction. $D_{\log m}$ has $m^{\log(m)}$ vertices but still has the property that no point in block k can be forced on before the k^{th} move. We define D_k below, algebraically, but please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

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



three blocks of D_3, respectively.

VERTICES(D_k) =

$$\left\{ \langle x_1,\ldots,x_k,r\rangle \mid r=b*k+p,\ p<k, b<2^k, 0\le x_i\le b+1 \text{ for } 1\le i\le p \ \&\ 0\le x_i\le b \text{ for } p<i\le k \right\}$$

EDGES(D_k) =

$$\{ (\langle x_1,\ldots,x_k,r\rangle, \langle x_1,\ldots,x_k,r+1\rangle) \mid r<k*2^k\} \quad \cup$$
$$\left\{ (\langle x_1,\ldots,x_k,r\rangle, \langle x_1,\ldots,x_{p-1},x_p+1,\ldots,x_k,r+1\rangle) \mid r=k*b + p \right\}$$

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 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 $j+1^{st}$ 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. 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 formula 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 n points. Thus to rember its orientation requires n bits rather than one. As above we can build graphs G_m' and H_m' which are m-equivalent without successor. We conjecture that even with Suc they are indistinguishable.

SECTION 4: Conclusions.

We have shown that quantifier rank is another measure of space complexity. Thus Ehrenfeucht games seem a likely tool for demonstrating lower bounds for space.

Furthermore the idea of quantifier rank unifies such notions as alternation and parallelism. We saw in the note after Theorem 1 that the device of alternating quantifiers is interchangeable with using an "and" or "or" to widen a formula without changing its depth. The latter device is intuitively identical to forking into two processors as in the Parallel RAM's of Savitch and Stimson [SaSt79].

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

1. We have seen that adding "Suc" allows formulas to count a bunch of identical points, and to keep track of the parity of binary switches. However, Theorem 1 suggests that "Suc" is not needed to express natural graph problems. Characterize those graph problems in NSPACE[log n] which are also in $QR[T_G, O(\log n)]$.

2. Is GAP still complete in some sense for all properties in $QR[T_G, O(\log n)]$? (The answer is probably "No," because Theorem 2 extends to $ALTSPACE[k, \log(n)] \subseteq QR[T^S, O(\log n)]$, where $ALTSPACE[k, \log(n)]$ is $\log(n)$ space for Turing machines allowed k alternations.) Using the notion of Interpretations Between Theories (see [End61] Section 2.7) we have defined reductions honoring QR. Via these reductions we have a natural complete set for $QR[f(n)]$ (when $f(n) < \sqrt{n}$), namely $E(f(n)) =$

$$\{\langle A,B\rangle \mid A,B \ f(n)\text{-equivalent graphs}\}.$$

3. Theorem 7 shows that although G_m and H_m differ on a P-time property, they are identical when only $(\log n)^k$ vertices are examined at once. We noted after the proof that two numbered graphs are $(\log(n)+1)$-equivalent only if they are identical. However it does not seem that a random numbering would provide a way of deciding reachablility. Perhaps we can develop a version of forcing on finite structures.

Conjecture: The short formulas forced by G_m' (i.e. true in all "generic" numberings) are the same as those forced by H_m'.

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.

[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.

[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.