JOURNAL OF COMPUTER AND SYSTEM SCIENCES 25, 76–98 (1982)

Upper and Lower Bounds for First Order Expressibility*

NEIL IMMERMAN

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

Received January 27, 1981; revised October 10, 1981

We study first order expressibility as a measure of complexity. We introduce the new class $\mathrm{Var\&Sz}[v(n), z(n)]$ of languages expressible by a uniform sequence of sentences with $v(n)$ variables and size $O[z(n)]$. When $v(n)$ is constant our uniformity condition is syntactical and thus the following characterizations of P and $PSPACE$ come entirely from logic.

$$NSPACE[\log n] \subseteq \bigcup_{k=1,2,\ldots} \mathrm{Var\&Sz}[k, \log(n)] \subseteq DSPACE[\log^2(n)],$$

$$P = \bigcup_{k=1,2,\ldots} \mathrm{Var\&Sz}[k, n^k],$$

$$PSPACE = \bigcup_{k=1,2,\ldots} \mathrm{Var\&Sz}[k, 2^{n^k}].$$

The above means, for example, that the properties expressible with constantly many variables in polynomial size sentences are just the polynomial time recognizable properties. These results hold for languages with an ordering relation, e.g., for graphs the vertices are numbered. We introduce an "alternating pebbling game" to prove lower bounds on the number of variables and size needed to express properties without the ordering. We show, for example, that k variables are needed to express Clique(k), suggesting that this problem requires $DTIME[n^k]$.

INTRODUCTION AND SUMMARY

This article studies the size and the number of variables of first order sentences needed to express certain properties. Each of these expressibility parameters measured individually is closely related to Turing machine space complexity. When variables and size are measured simultaneously they correspond to simultaneous space and time. Thus the computational complexity of testing if an input has a given property can be measured by determining the size and number of variables needed to express the property in the language of mathematics. This insight suggests new ways to obtain upper and lower bounds on a problem's complexity. Upper bounds are

[1] Research partly supported by NSF Grant MCS 81-05754.

0022-0000/82/040076–23$02.00/0

obtained by expressing the property succinctly. Lower bounds can be demonstrated by showing that two structures which differ on the property in question agree on all sentences of a certain size and containing a certain number of variables.

In [12] we proposed studying the complexity of a property, C, via the size of a sentence from first order logic needed to express C. We showed there that the memory space needed to check if a given input has property C is closely related to the size of C's smallest first order description. More precisely:

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

Here $\text{Size}[g(n)]$ is the family of all properties expressible by a uniform sequence of sentences, $F_1, F_2, \ldots$, where F_n has $O[g(n)]$ symbols.

Several papers by Ruzzo [19, 20], on simultaneous resource bounds motivated us to find analogous results for first order expressibility. First we reexamined our proof of the above containment for $f(n) = \log(n)$, i.e.:

$$NSPACE[\log(n)] \subseteq \text{Size}[\log(n)] \subseteq DSPACE[\log^2(n)]$$

and noticed that only a constant number of variables were needed. Furthermore while the existential quantifiers range over the elements of the universe of the input, (i.e., 1 to n), the universal quantifiers could be boolean. Thus we let $Var\&Sz[v(n), z(n)]$ be the class of properties uniformly expressible with exactly $v(n)$ variables and size $O[z(n)]$. Also let $\text{Var\&Sz}(B\forall)[v(n), z(n)]$ be the same class with the additional restriction that the universal quantifiers are boolean. Let "$*$" abbreviate $O[1]$. We show that:

$$NSPACE[\log(n)] \subseteq \text{Var\&Sz}(B\forall)[*, \log(n)] \subseteq \text{Var\&Sz}[*, \log(n)]$$
$$\subseteq \text{Size}[\log(n)] \subseteq DSPACE[\log^2(n)].$$

Although none of the containments above are known to be proper, we conjecture that all four are. Savitch's simulation of $NSPACE[\log(n)]$ by $DSPACE[\log^2(n)]$ may be optimal, but a model theoretic approach to separating the two classes would be to prove

$$\text{Var\&Sz}(B\forall)[*, \log(n)] \neq \text{Size}[\log(n)].$$

We find that $\text{Var\&Sz}(B\forall)[* \log(n)]$ is identical to the natural class Log(CFL)—those languages log-space reducible to some context free language. We will also see that the third term in the above chain, $\text{Var\&Sz}[*, \log(n)]$, is equal to $ASPACE\&Alt[\log(n), \log(n)]$—the class of languages accepted by an $ASPACE[\log(n)]$ Turing machine which makes only $O[\log(n)]$ alternations between existential and universal states.

Once the idea of counting distinct variables was raised it was natural to relax the size restriction. Let $\text{Var}[*, n^*] = \bigcup_{k=1,2,\ldots} \text{Var\&Sz}[k, n^*]$—those properties expressible with a constant number variables in polynomial size sentences $\text{Var}[*, n^*]$ is identical to polynomial time!

One weakness of our previous definition of expressibility size is that it makes use of the notion of Turing machines in the definition of a "uniform" sequence of sentences. At the time our feeling was that the uniformity condition was an imperfect attempt to capture the notion that we really had one sentence with a variable number of quantifiers, just as we have the notion of one Turing machine with a variable amount of space. Indeed, the use of constantly many variables leads us to the realization that there is a syntactic uniformity—the nth sentence of a Var&Sz$[k, z(n)]$ property is just $z(n)$ repetitions of a fixed block of k quantifiers. With this new definition of uniformity, Var$[*, n^*]$ is a notion entirely from logic.

Now that we know that $DTIME[n^k]$ is closely related to Var$[*, n^*]$ it is useful to determine which graph properties can and cannot be expressed with k variables. In Section C we describe a combinatorial game, a modification of Ehrenfeucht–Fraisse games, (see [6] or [10]), with which we can prove lower bounds on what can be expressed in k variables. These new games are an alternating version of pebbling games.

Our definition of Var&Sz gives the sentences access to some arbitrary ordering relation, $\leqslant$, on the universe of the input structures. Without this added relation we cannot simulate Turing machines—there is no way to say, "Now the Turing machine moves its input head one space to the right." We showed in [12] that $\leqslant$ is not needed to express certain "natural" graph problems such as connectivity; however, it is essential for other uses such as counting the parity of a totally disconnected graph.

The games mentioned above give us lower bounds only on what can be expressed without the ordering predicate. We show, for example, that Clique(k)—the existence of a complete subgraph on k vertices—cannot be expressed with $k-1$ variables, without $\leqslant$. (Of course k variables suffice—just say there exist $x_1 \cdots x_k$ forming a clique.) This is plausability argument that Clique(k) is not in Var$[k-1, n^*]$. If we could prove the latter result, i.e., that Clique(k) cannot be expressed with $k-1$ variables and polynomial size in the language with $\leqslant$, then it would follow that the general clique problem is not in Var$[*, n^*]$. From this it would follow that $P \neq NP$.

In the following pages we give: (A) Definitions and motivations; (B) Some of the main relationships between expressibility and Turing machine Time and Space; (C) The alternating pebbling game; (D) Probabilistic graph arguments following [8, 2] showing that Hamilton Circuit, Clique, and GraphIso are not in Var(w.o. $\leqslant$)$[*]$; and (E) Conclusions and directions for future research.

A. Definitions and Motivations

We propose to study the complexity of a condition, C, by asking, "How difficult is to express C?" For this expression we choose the natural first order language of the objects under consideration.

Think of a directed graph, for example, as a universe, $V = \{0, 1, \ldots, n-1\}$, the vertices, together with a binary edge relation $E(-,-)$ on V. This is a logical *structure* of *similarity type* $\tau_G = \langle E(-,-)\rangle$. The language of a type, τ, $L[\tau]$, consists of the sentences built up from the symbols of τ using the logical connectives &, "or", $\neg$, $\Rightarrow$, variables x, y,..., $=$, $\leqslant$ and quantifiers, $\exists x$ and $\forall x$, ranging over the universe. The two relations, $=$ and $\leqslant$, refer to the equality relation and the natural ordering on the universe. For example, consider the following sentence from $L[\tau_G]$:

$$S_1 \equiv \forall x\, \exists y [E(x, y) \text{ or } E(y, x)].$$

S_1 says that each vertex, x, has an edge coming out of it or an edge going into it. A graph *satisfies* S_1 (in symbols $G \models S_1$) if it has no isolated vertices. Note that every graph G "understands" every sentence S from $L[\tau_G]$, i.e., $G \models S$ or $G \models \neg S$.

To motivate the definitions for variable and size expressibility we now consider a stepwise refinement of sentences expressing a specific problem. Let GAP be the set of directed graphs G with specified points a and b such that there is a path in G from a to b. In symbols:

$$GAP = \{G \mid a \to * \to b\}.$$

GAP is known to be complete for $NSPACE[\log(n)]$. (See [21].) We show in [13] that GAP is complete in a very strong sense—every problem C in $NSPACE[\log(n)]$ has a first order sentence translating all instances of C into instances of GAP.

To express GAP we will write down formulas $P_n(a, b)$ meaning, "There is a path of length at most n from a to b." We define P_n by induction as follows:

$$P_1(x, y) \equiv (x = y) \text{ or } E(x, y), \tag{1}$$

$$P_n(x, y) \equiv \exists z (P_{n/2}(x, z) \& P_{n/2}(z, y)). \tag{2}$$

Equation (2) defines P_n in a way that increases the quantifier rank, i.e., maximum nesting of quantifiers, by one each time n is doubled. However $P_{n/2}$ is written twice on the right so the size of this P_n is twice the size of $P_{n/2}$. We can alleviate this problem using the "abbreviation trick" (see, e.g., [9]). The trick uses universal quantifiers to permit us to write $P_{n/2}$ only once on the right. Thus:

$$P_n(x, y) \equiv \exists z\ \forall u\ \forall v([u = x \& v = z \text{ or } u = z \& v = y] \Rightarrow P_{n/2}(u, v)). \tag{3}$$

We have now written P_n with $O[\log(n)]$ symbols, thus proving that GAP is in $\mathrm{Size}[\log(n)]$, to be defined.

Continuing in our refinement notice that when we write $P_{n/2}(u, v)$ we may reuse x, y, z—their current values are no longer needed. Being slightly wasteful for the sake of clarity, write:

$$P_n(x, y) \equiv \exists z\, \forall u\, \forall v([u = x \& v = z \text{ or } u = z \& v = y] \Rightarrow \exists x\, \exists y [x = u \& y = v \& P_{n/2}(x, y)]). \tag{4}$$

We have succeeded in expressing GAP by a uniform sequence of sentences,

$\{P_n(a, b) \mid n \geqslant 1\}$, such that P_n has five variables and size $O[\log(n)]$. This suggests the following:

DEFINITION. A set C of structures of type τ is *expressible in $v(n)$ variables and size $z(n)$*, (in symbols, C is in *Var&Sz*$[v(n), z(n)]$), if there exists a *uniform* sequence of sentences $F_1 F_2 \ldots$ from $L(\tau)$ such that:

a. For all structures G of type τ with $|G| \leqslant n$,

$$G \in C \leftrightarrow G \vDash F_n$$

b. F_n has $v(n)$ distinct variables and a total of $O[z(n)]$ symbols.

As Ruzzo has shown in [20], uniformity conditions may be greatly varied without significantly changing a definition. The following condition will suffice in what follows:

Uniformity Condition (*): The map $n \rightarrow F_n$ can be generated in $DSPACE[v(n) \cdot \log(n)]$ and $DTIME[z(n)]$.

Of course (*) does not capture our intuitive feeling that the F_n's are all the same sentence with varying numbers of quantifiers. To make the latter notion more precise abbreviate quantifiers with restricted domains as follows:

$$(\exists x \cdot M)[\ \cdots\ ] \equiv z[M\& \ \cdots\ ] \qquad \text{read, "There exists } x \text{ such that } M."$$

$$(\forall x \cdot M)[\ \cdots\ ] \equiv \forall x[M \Rightarrow \ \cdots\ ] \text{ read, "For all } x \text{ such that } M."$$

Now we can write Eq. (4) more compactly as:

$$P_n(x, y) \equiv \exists z\ \forall u(\forall v \cdot M_3)\ \exists x(\exists y \cdot M_5)\ P_{n/2}(x, y). \tag{5}$$

Here $M_3 = [u = x\&v = z$ or $x = z\&v = y]$, and $M_5 = [x = u\&y = v]$. Let $A \equiv (\exists x \cdot x = a)(\exists y \cdot y = b)$. We can now write the sentences GAP_n expressing the existence of a path of length at most n from a to b in a very neat form:

$$GAP_n \equiv A[(\exists z)(\forall u)(\forall v \cdot M_3)(\exists x)(\exists y \cdot M_5)]^{\log(n)} P_1. \tag{6}$$

Equation (6) give a model for the following totally syntactical definition of uniformity for Var&Sz$[v, z(n)]$:

Uniformity Condition (**): There exist constant c, prefix A, and quantifier free formulas B, $M_1 \cdots M_v$ all of which have variables only $x_1 \cdots x_v$ such that:

$$F_n \equiv A[(Q_1 x_1 \cdot M_1) \cdots (Q_v x_v \cdot M_v)]^{c \cdot z(n)} B.$$

We adopt (**) as our definition of uniformity for Var&Sz$[v, z(n)]$ when v is a constant, otherwise we use (*). Equation (6) demonstrates that GAP is in Var&Sz$[5, \log(n)]$. More generally we can show:

THEOREM A.1. (a) *For* $s(n) > O[\log(n)]$,

$$NSPACE[s(n)] \subseteq \text{Var\&Sz}[O[s(n)/\log(n)], s(n)^2/\log(n)] \subseteq DSPACE[s(n)^2].$$

(b):

$$NSPACE[\log(n)] \subseteq \text{Var\&Sz}[*, \log(n)] \subseteq DSPACE[\log^2(n)].$$

Proof. The proof of (a) is nearly the same as for Theorem 2 in [15]. We showed there that $NSPACE[s(n)] \subseteq \text{Size}[s(n)^2/\log(n)] \subseteq DSPACE[s(n)^2]$. That proof noted that a Turing machine instantaneous description (ID) of size $s(n)$ could be coded in $O[s(n)/\log(n)]$ variables since the variables range over an n element universe. Thus using Eq. (3) we asserted the existence of a computation path of length $c^{s(n)}$; $O[s(n)]$ ID's were needed. For the proof of the first inclusion in (a) we use Eq. (4) instead. Thus only a constant number of ID's, requiring $O[s(n)/\log(n)]$ variables, must be remembered at once.

Part (b) seems to be a special case of (a) but the proof of the first inclusion is more subtle because we must satisfy the syntactic uniformity condition (**). The following proof is quite technical and may easily be skipped at first reading without affecting the understanding of the remainder of the paper.

We are given a nondeterministic Turing machine M running in space $\log(n)$ and accepting a subset of all the structures of some type. Assume for convenience that M accepts a set of graphs, i.e., $\tau = \langle E(-,-)\rangle$, and that the inputs are adjacency matrices. We must build sentences φ_n expressing the acceptance property of M for graphs of size n. Furthermore the φ_n's must be syntactically uniform, have constantly many variables, and be of size $O[\log(n)]$.

Since the variables range over an n-element universe they may be thought of as $\log(n)$ bits of memory. We can thus code M's $\log(n)$ size instantaneous description (ID) with a constant number of variables. An ID is coded as $\langle q, r_1, r_2, w_1 \cdots w_k, h_1 \cdots h_k\rangle$, where q gives the state and $w_1 \cdots w_k$ and $h_1 \cdots h_k$ code the $k(\log(n))$ bits of work tape, and the position of the work head, respectively. Finally, r_1 and r_2 encode the read head position, i.e., they indicate that the read head is looking at the cell corresponding to the pair $\langle r_1, r_2\rangle$ in the adjacency matrix. Thus the read head is looking at a 1 if $E(r_1, r_2)$ holds for the input graph, otherwise it is looking at a 0. Note that the ordering $\leqslant$ is used to indicate that the read head moves one space to the left or right. This is the crucial use of $\leqslant$. A less important use is to code and decode $\log(n)$ bits as a single variable.

It is a small matter to recognize M's initial and final ID's. We will show how to write the formula $P_1(ID_a, ID_b)$ meaning that ID_b follows from ID_a in one move of M. We then use Eq. (6) to express $P_{n^k}(ID_i, ID_f)$, that there is a computation path of length n^k from M's initial ID to M's final ID.

To write P_1 we must be able to say, "The symbol being read by the work head is 0." It is thus necessary to decode the ith bit of a vertex's number. We will identify a vertex with its number. Let $ON_n(x, y)$ mean that $y < \log(n)$ and bit y of x is a 1.

LEMMA A.2. *$ON_n(x, y)$ may be written uniformly in* Var&Sz[*, log(n)].

Proof. We build up to $ON_n(x, y)$ using a sequence of inductive definitions and repeatedly using an abbreviation trick as in [9]. We will use the symbols 0 and 1 for convenience but they are of course definable from $\leqslant$. Define the successor relation by

$$\mathrm{Suc}(x, y) \equiv (x \leqslant y) \,\&\, (x \neq y) \,\&\, (\forall z)[(x \leqslant z) \Rightarrow (z = x \text{ or } y \leqslant z)].$$

(a) Define $\mathrm{Plus}_n(x, y, z)$ to mean $(x \leqslant n)$ and $x + y = z$:

$$\mathrm{Plus}_1(x, y, z) \equiv (x = 0 \,\&\, y = z) \text{ or } (x = 1 \,\&\, \mathrm{Suc}(y, z)).$$

$$\mathrm{Plus}_{2k}(x, y, z) \equiv \exists u_1 \exists u_2 \exists u_3 (\mathrm{Plus}_k(u_1, u_2, x) \,\&\, \mathrm{Plus}_k(u_1, y, u_3) \,\&\, \mathrm{Plus}_k(u_2, u_3, z)).$$

Using the abbreviation trick:

$$\mathrm{Plus}_{2k}(x, y, z) \equiv \exists u_1 \exists u_2 \exists u_3 \forall s_1 \forall s_2 (\forall s_3 \cdot [(s_1 = u_1 \& s_2 = u_2 \& s_3 = x) \text{ or } (s_1 = u_1 \& s_2 = y \& s_3 = u_3) \text{ or } (s_1 = u_2 \& s_2 = u_3 \& s_3 = z)])\, \mathrm{Plus}_k(s_1, s_2, s_3).$$

Let

$$A_1 \equiv [(s_1 = u_1 \& s_2 = u_2 \& s_3 = x) \text{ or } (s_1 = u_1 \& s_2 = y \& s_3 = u_3) \text{ or } (s_1 = u_2 \& s_2 = u_3 \& s_3 = z)]$$

and

$$A_2 \equiv (x = s_1 \,\&\, y = s_2 \,\&\, z = s_3).$$

We thus obtain a syntactically uniform form of Plus:

$$\mathrm{Plus}_n(x, y, z) \equiv [\exists u_1 \exists u_2 \exists u_3 \forall s_1 \forall s_2 (\forall s_3 \cdot A_1)\, \exists x \exists y (\exists z \cdot A_2)]^{\log(n)} P_1(x, y, z).$$

(b) Define $M_n(p, q, r)$ to mean $(p \leqslant n \,\&\, r \leqslant n \,\&\, pq = r)$:

$$M_1(p, q, r) \equiv (p = r = 0) \text{ or } (p = 1 \,\&\, q = r \,\&\, (r = 0 \text{ or } r = 1)).$$

$$M_{2k}(p, q, r) \equiv \exists u_1 \exists u_2 \exists u_3 \exists w_1 \exists w_2 (M_k(u_1, q, w_1) \,\&\, M_k(u_2, q, w_2) \,\&\, \mathrm{Plus}_k(u_1, u_2, p) \,\&\, \mathrm{Plus}_k(w_1, w_2, r)).$$

This definition works because $p = u_1 + u_2$, and so $pq = u_1 q + u_2 q$. To put M_n into syntactically uniform we need a lemma.

LEMMA A.3 "Combining Lemma". *Suppose that $A_n(x)$ can be written in syntactically uniform form*

$$A_n(x) \equiv [ABLOCK]^{\log n} A_0(x),$$

and suppose that B_n may be defined inductively as

$$B_{2m}(y) \equiv (Q_1 y_1) \cdots (Q_k y_k) R(y, A_m, B_m),$$

where R is a quantifier free formula. Then B_n may be written in the syntactically uniform form,

$$B_n(x) \equiv [BBLOCK]^{\log n} B_0(x).$$

Proof. We must combine occurrences of A_m and B_m in R into a single occurrence of a formula C_m of size $O[\log m]$. $C_m(\alpha, x, y)$ will be equivalent to:

$$(\alpha = 0 \Rightarrow A_m(x)) \;\&\; (\alpha = 1 \Rightarrow B_m(y)).$$

Thus an inductive definition for C_m is:

$$\begin{aligned} C_0(\alpha, x, y) &\equiv (\alpha = 0 \Rightarrow A_0(x)) \;\&\; (\alpha = 1 \Rightarrow B_0(y)), \\ C_{2m}(\alpha, x, y) &\equiv [ABLOCK](Q_1 y_1) \cdots (Q_k y_k)((\alpha = 0 \Rightarrow C_m(0, x, y)) \\ &\qquad \& \; (\alpha = 1 \Rightarrow R(y, C_m(0, -, y), C_m(1, x, -))), \\ &\equiv [Q\text{-}B\text{-}L\text{-}O\text{-}C\text{-}K](P(\alpha, x, y, C_m)). \end{aligned}$$

Here $QBLOCK$ is a quantifier block and P is a quantifier free formula. By induction $C_m(0, -, y)$ is equivalent to $A_m(-)$, and $C_m(1, x, -)$ is equivalent to $B_m(-)$. We will assume that all occurrences of A_m and B_m in R are positive. Thus so are the occurrences of C_m in P. If this were not true then we would expand C_m to include such cases as $(\alpha = 2 \Rightarrow \neg A_n(x))$. Assume that P is in disjunctive normal form, i.e.,

$$P(\alpha, x, y, C_m) \equiv \bigvee_{i = 1 \cdots r} \; \bigwedge_{j = 1 \cdots \max_i} F_{ij},$$

where each F_{ij} is either $C_m(b_{ij}, x_{ij}, y_{ij})$, or τ_{ij}—a quantifier free formula not involving C_m. A formula equivalent to $P(\alpha, x, y, C_m)$ is,

$$L(\alpha, x, y, C_m) \equiv (\exists i \cdot 1 \leqslant i \leqslant r)(\forall j)(\exists \beta u v \cdot S)\, C_m(\beta, u, v).$$

Here S is a conjunction over i and j saying that $C_m(\beta, u, v)$ is equivalent to F_{ij}. In the case where F_{ij} is $C_m(b_{ij}, x_{ij}, y_{ij})$ we must assert that $\beta = b_{ij}$, $u = x_{ij}$, and $v = y_{ij}$. If F_{ij} is τ_{ij}, we merely assert that τ_{ij} holds. In symbols,

$$S \equiv [(i = 1 \;\&\; j = 1) \Rightarrow T_{11}] \;\&\; \cdots \;\&\; [(i = r \;\&\; j = \max_r) \Rightarrow T_{r \max_r}],$$

where

$$\begin{aligned} T_{ij} &\equiv [\beta = b_{ij} \;\&\; u = x_{ij} \;\&\; v = y_{ij}] && \text{if } F_{ij} \text{ is } C_m(b_{ij}, x_{ij}, y_{ij}) \\ &\equiv \tau_{ij} && \text{if } F_{ij} \text{ is } \tau_{ij}. \end{aligned}$$

$L(\alpha, x, y, C_m)$ is clearly equivalent to $P(\alpha, x, y, C_m)$. Thus,

$$\begin{aligned} C_{2m}(\alpha, x, y) &\equiv [QBLOCK]\, L(\alpha, x, y, C_m) \\ &\equiv [QBLOCK](\exists i \cdot 1 \leqslant i \leqslant r)(\forall j)(\exists \beta uv \cdot S)\, C_m(\beta, u, v) \\ &\equiv [QBLOCK](\exists i \cdot 1 \leqslant i \leqslant r)(\forall j)(\exists \beta uv \cdot S)(\exists \alpha xy \cdot E)\, C_m(\alpha, x, y) \\ &\equiv [CBLOCK]\, C_m(\alpha, x, y). \end{aligned}$$

Here E says $(\beta = \alpha \& u = x \& v = y)$. Thus we have written C_n and thus B_n in the form desired:

$$C_n \equiv [CBLOCK]^{\log n}\, C_0. \quad \blacksquare$$

(c) Define $EXP_n(a, b)$ to mean $(a \leqslant \log(n))\ \&\ (2^a = b))$:

$$EXP_1(a, b) \equiv (a = 0\ \&\ b = 1),$$
$$EXP_{2k}(a, b) \equiv (\exists c\, d)(EXP_k(c, d)\ \&\ \mathrm{Suc}(c, a)\ \&\ \mathrm{Plus}_k(d, d, b)).$$

(d) Define $ON_n(x, y)$ to mean $(y \leqslant \log(n))$ & (the yth bit of x is a 1):

$$ON_n(x, y) \equiv (\exists u\, v\, w\, t)(EXP_n(y, v)\ \&\ u + v + w = x\ \&\ M_n(v, t, w)).$$

By the combining lemma, EXP_n and ON_n can be written in a syntactically uniform form. This proves Lemma A.2. $\blacksquare$

Now that we have $ON_n(x, y)$ we can write $P_1(ID_a, ID_b)$. P_1 is just a disjunction over all triples of states and read and work symbols saying what the Turing machine M will do in one step. Then as claimed we can write $P_{n^k}(ID_i, ID_f)$, expressing the acceptance property for M uniformly with $O[1]$ variables and size $O[\log(n)]$. $\blacksquare$

Let's return to Eq. (4) and notice that in simulating an $NSPACE[\log(n)]$ property, two universal quantifiers ranging from 1 to n are used. Their purpose is only to make a choice between the first half and the second half of the path. It makes sense to minimize the universal choices when simulating an existential class so we replace "$\forall u \forall v$" in Eq. (4) by "$\forall b$", where b is boolean valued. Thus:

$$\begin{aligned} P_n(x, y) \equiv \exists z \forall b \exists u [\exists v \cdot [(b = 0 \& u = x \& v = z) \text{ or } (b = 1 \& u = z \& v = y)]) \\ \exists x (\exists y \cdot [x = u \& y = v))\, P_{n/2}(x, y). \end{aligned} \tag{7}$$

Define $\mathrm{Var\&Sz}(B\forall)[v(n), z(n)]$ to be the family of properties expressible in $v(n)$ variables and size $O[z(n)]$, where the existential quantifiers still range from 1 to n, but the universal quantifiers are boolean. We will always assume that $z(n) \geqslant O[\log n]$, and allow the sentences in question to contains constantly many ordinary universal quantifiers, i.e., $O[\log n]$ bits. This is useful for example in defining 0 and 1 which are needed in the proof of Theorem A.1. For the definition to make sense we assume that the formulas are in prenex form with all the $\neg$'s pushed inside. It is easy to see that GAP is in Var&Sz$(B\forall)[k, \log(n)]$, and more generally,

THEOREM A.4. *For* $s(n) \geqslant \log(n)$,

$$NSPACE[s(n)] \subseteq \text{Var\&Sz}(B\forall)[O[s(n)/\log(n)], s(n)^2/\log(n)].$$

B. VARIABLES & SIZE VERSUS TIME & SPACE

Recall a definition and result of Sudborough [22]:

DEFINITION. $AuxPDA[s(n), t(n)]$ is the class of languages accepted by a two way nondeterministic push down automaton with auxiliary work tape of size $s(n)$, running in time $t(n)$.

FACT (Sudborough). $AuxPDA[\log(n), n^*] = \log(CFL)$.

Ruzzo [19] defines an *accepting computation tree* of an alternating Turing machine M to be a tree whose root is a starting ID of M, whose nodes are intermediate ID's and whose leaves are accepting configurations. Each universal node, u, has all its possible next moves as offspring, while the existential nodes, e, lead to exactly one of e's possible next moves. We say that a language C is in $ASPACE\&TS[s(n), z(n)]$ if all members of C of size n are accepted in a computation tree using space $s(n)$ and tree size, (number of nodes), $z(n)$. Ruzzo relates this new measure to auxiliary pda's via his Theorem 1 which implies:

FACT (Ruzzo). $ASPACE\&TS[s(n), z(n)^*] = \text{Aux}PDA[s(n), z(n)^*]$.

Notice that both the tree size model and the AuxPDA charge much more for universal moves than for existential ones. The following theorem shows that we get the same classes in our expressibility measure by restricting all universal quantifiers to be boolean. In a sense we charge $\log(n)$ times as much for a universal choice as for an existental one.

THEOREM B.1.

$$\begin{aligned}\text{Var\&Sz}(B\forall)[O[v(n)], z(n)] &\subseteq ASPACE\&TS[v(n)\log(n), {*}^{z(n)}]\\ &\subseteq \text{Var\&Sz}(B\forall)[O[v(n)], v(n)\cdot z(n)].\end{aligned}$$

Proof. $(\subseteq_1)$: Given an input structure G with n element universe we can generate F_n, the nth sentence in our uniform sequence. We must show that in $ASPACE\&TS[v(n)\log(n), {*}^{z(n)}]$ we can check if $G \models F_n$.

To test if G satisfies F_n we read the sentence from left to right holding the present values of variables $x_1 \cdots x_{v(n)}$ in our $v(n)\log(n)$ memory. Note that each nonboolean variable may have value 1 to n corresponding to an element of G. At existential quantifiers, $\exists x_i$, we existentially choose some x_i from the universe of G and at universal choices, $\forall b_j$, we universally choose b_j. When we come to atomic predicates, e.g., $E(x_1, x_2)$ or $b_{17} = 0$, we can check their truth because we have the current values of the variables. Note that this accepting procedure has tree size $*^{z(n)}$ because we may make a binary universal split $O[z(n)]$ times.

($\subseteq_2$): Here we follow a proof of Ruzzo [19]. We must express the property Accept(r, z) which means that the alternating Turing machine M will accept in tree size z when started with ID r. We express Accept(r, z) by choosing a point p in the middle of the tree whose subtree is of size between 1/3 and 2/3 of the original tree. We may assume that the alternating machine has at most two choices at each move. Thus it is obvious that such a p exists. Thus,

$$\text{Accept}(r, z) \equiv \exists p(\text{Accept}(r, \langle p \rangle, (2/3)z) \;\&\; \text{Accept}(p, \langle\, \rangle, (2/3)z)).$$

Here Accept$(r, \langle q_1 \cdots q_k \rangle, z)$ means that there is a computation tree of size z starting at r such that each leaf is either an accepting configuration or one of $q_1 \cdots q_k$.

Our only trouble is to ensure that the list $\langle q_1 \cdots q_k \rangle$ stays of constant size. Whenever the list is of length three we take an extra move to split it in half by finding a point p above two of the three nodes in the list,

$$\text{Accept}(r, \langle q_1, q_2, q_3 \rangle, z) \equiv \exists p(\text{Accept}(r, \langle q_1, p \rangle, z) \;\&\; \text{Accept}(p, \langle q_2, q_3 \rangle, z)).$$

Note that in the above we can add a boolean universal quantifier and use the abbreviation trick to write Accept(–) only once on the right. Also note that the above is a slight lie since we don't know which pair of q's p will be above. In fact we would have to say,

$$\exists p(\exists s_1, s_2, s_3, \text{ a permutation of } q_1 q_1 q_3)(\text{Accept}(r, \langle s_1, p), z) \;\&\; \text{Accept}(p, \langle s_2, s_3 \rangle, z)).$$

Thus we can write Accept(–) with a constant number of ID's, i.e., $O[v(n)]$ variables, and the size of the sentence is $O[v(n) \cdot \log(z)]$. This proves Theorem B.1. ∎

COROLLARY B.2. $\text{Var\&Sz}(B\forall)[* \log(n)] = \text{Log}(CFL)$.

Proof. From the above theorem, together with the results from Ruzzo and from Sudborough:

$$\begin{aligned}\text{Var\&Sz}(B\forall)[*, \log(n)] &= ASPACE\&TS[\log(n), n^*] \\ &= \text{Aux}PDA[\log(n), n^*] \\ &= \text{Log}(CFL). \quad ∎\end{aligned}$$

There is a close relationship between expressibility and the complexity of alternating Turing machines [3, 16]. As we see in the next theorem the number of variables corresponds to alternating space while the size of the sentence is similar to alternating time.

THEOREM B.3. *For* $s(n) \geqslant \log(n)$,

$$ASPACE\&TIME[s(n), t(n)] \subseteq \mathrm{Var\&Sz}[O[s(n)/\log(n)], t(n)] \subseteq ASPACE\&TIME[s(n), t(n)\log(n)].$$

Proof. $(\subseteq)_1$ Given an alternating machine M, we must write sentences φ_n so that an input G of size n is accepted by M if and only if G satisfies φ_n. We write the sentences $\text{Accept}_t(x)$ to mean that M started at IDx will accept within t steps. We accomplish this by saying that if x is in an existential state then there is some next IDy such that $\text{Accept}_{t-1}(y)$ whereas if x is universal then all next ID's y satisfy $\text{Accept}_{t-1}(y)$.

$$\text{Accept}_t(x) \equiv (\exists y)(P_1(x, y) \,\&\, \text{Accept}_{t-1}(y)) \,\&\, (\text{“}x \text{ is universal”} \Rightarrow [(\forall y)(P_1(x, y) \Rightarrow \text{Accept}_{t-1}(y)]).$$

Here x and y code ID's of the form $\langle q, x_1, h_1, x_2, h_2, ..., x_r, h_r\rangle$ where $r = O[s(n)/\log(n)]$, q is the state, $x_1 \cdots x_r$ hold the tape contents, and all the h_i's are 0 except one indicating the location of the head at a cell, $1 \leqslant h_i \leqslant \log(n)$ of x_i. $P_1(x, y)$ is as in the proof of Theorem A.1; it means that IDy follows from IDx in one move of M. First we rewrite Accept_t using the abbreviation trick:

$$\text{Accept}_t(x) \equiv (\exists y)(\forall z)(P_1(x, y) \,\&\, [z = y \text{ or } (\text{“}x \text{ is universal”} \,\&\, P_1(x, z))] \Rightarrow \text{Accept}_{t-1}(z))$$

We have written Accept_t using $O[t]$ blocks of quantifiers where each block quantifies the $O[s(n)/\log(n)]$ variables needed to code one ID. If $s(n) > [\log n]$ then this is wasteful because the whole ID need not be requantified at each step. We will sketch why it suffices to requantify two adjacent pieces of the tape at each move, and to requantify the whole ID only once every r steps, thus keeping the formula size linear in $t(n)$. Let symbols a and b abbreviate 6-tuples of the form $\langle q, x_i, h_i, x_{i+1}, i\rangle$ representing the state and the ith and $i+1$st pieces of an instantaneous description. The idea is to requantify such 6-tuples rather than the whole ID at each move. Thus,

$$\text{Accept}_t(x) \equiv (\exists a)(\forall b)(P_1(x, a) \,\&\, [b = a \text{ or } (\text{“}x \text{ is universal”} \,\&\, P_1(x, b))] \Rightarrow \text{Accept}_{t-1}(\langle x, b\rangle)).$$

In order to repeat r blocks of these a's and b's it is convenient to use extra variables c to keep track of what is going on. Let c_i be 0 if $a_i = b_i$ or if b_{i-1} is universal and b_i is a possible next move after IDx followed by $b_1 \cdots b_{i-1}$, in symbols $P_1(\langle x, b_1 \cdots b_{i-1}\rangle, b_i)$. Let c_i be 1 otherwise. Thus:

$$\text{Accept}_t(x) \equiv (\exists a)(\forall b)(\exists c)([c=0 \;\&\; \text{Accept}_{t-1}(\langle x, b\rangle)] \text{ or }$$
$$(c=1 \;\&\; a \neq b \;\&\; (\text{“}x \text{ is existential” or } \neg P_1(x, b))]).$$

Once each r steps a whole new IDx' is requantified:

$$\text{Accept}_t(x) \equiv (\exists a_1)(\forall b_1)(\exists c_1)(\exists a_2)(\forall b_2)(\exists c_2) \cdots (\exists a_r)(\forall b_r)(\exists c_r)(\exists x')$$
$$(M \text{ or } [N \;\&\; \text{Accept}_{t-r}(x')]).$$

Here M says "$(\exists k)\, c_1 = c_2 = \cdots = c_{k-1} = 0$ and $x \to b_1 \to \cdots \to b_{k-1}$ codes $k-1$ steps of a valid computation of M, and $c_k = 1$ and c_{k-1} is existential or $\neg P_1(\langle x, b_1 \cdots b_{k-1}\rangle, b_k)$." N says, "$c_1 = c_2 = \cdots = c_r = 0$ and $x \to b_1 \to \cdots \to b_r$ codes r steps of a valid computation of M resulting in x'."

M and N can be written with $O[r]$ symbols, using predicate P_1 and a new predicate $T(x, b_1, \ldots, b_r, i, j, w)$ meaning that at step i, $1 \leqslant i \leqslant r$, the contents of section j of the ID is w. Finally, as in the combining lemma we can combine P, T, and Accept into one predicate which can be defined as a formula of length $O[t(n)]$ using $O[s(n)/\log(n)]$ variables. The desired φ_n is $\text{Accept}_{t(n)}(ID_i)$, where ID_i is M's initial instantaneous description.

$(\subseteq)_2$: Here we must show that given a structure G of size n and a sentence φ_n with $s(n)/\log(n)$ variables and size $t(n)$ we can check in $ASPACE\&TIME[s(n), t(n)\log(n)]$ whether or not G satisfies φ_n. To test if G satisfies φ_n we read the sentence from left to right holding the present values of the variables $x_1 \cdots x_{s(n)/\log(n)}$ in our $s(n)$ bit memory. At quantifiers $(\exists x_i)$ or $(\forall x_i)$ we make the appropriate existential or universal choice of a new value for x_i. Similarly at &'s or "or"s we can universally or existentially choose one branch and proceed. The atomic sentences can be checked in constant time assuming we are dealing with indexing alternating Turing machines. Note that this simulation requires up to $\log(n)$ steps for each symbol of φ_n. ∎

Theorem B.3 would be nicer if we could improve the size bound in the middle term to $t(n)/\log(n)$. This seems unlikely however, because the alternating time $t(n)$ machine can make $t(n)$ alternations while the sentence could make only $t(n)/\log(n)$ alternations. We can get an exact relation between expressibility and alternating complexity by restricting the number of alternations the machine may make.

THEOREM B.4. *For $s(n) \geqslant \log(n)$,*

(a) $ASPACE\&TIME\&Alt[s(n), t(n), t(n)/\log(n)]$
$= \text{Var\&Sz}[O(s(n)/\log(n)], t(n)/\log(n)]$.

(b) $ASPACE\&Alt[s(n), a(n)]$
$\subseteq \text{Var\&Sz}[O[s(n)/\log(n)], a(n) + s(n))\, s(n)/\log(n)]$.

(c) $ASPACE\&Alt[\log(n), \log(n)] = \text{Var\&Sz}[*, \log(n)]$.

Proof sketches: (a) Here the class on the left consists of languages recognizable by an alternating Turing machine simultaneously in space $s(n)$, time $t(n)$, and making $t(n)/\log(n)$ alternations. The proof is similar to the proof of Theorem B.3. The difference is that when the next $\log(n)$ steps involve no alternation we skip ahead $\log(n)$ steps with one quantifier and check later that all such jumps were valid $\log(n)$ step computations. In this way we use a constant number of quantifiers for each alternation and for each $\log(n)$ moves.

(b) The proof of (b) is similar except that we have no time bound between alternations. Thus we must write out the whole *ID*, i.e., $O[s(n)/\log(n)]$ variables, at the endpoints of each of the $a(n)$ alternations. We check once that within an alternation the final *ID* follows from the initial *ID*. By Theorem A.1 this may be expressed with $s(n)^2/\log(n)$ symbols.

(c) One half of (c) is a special case of (b). The other half is similar to the second containment of Theorem B.3. Evaluating a sentence with $O[1]$ variables and size $O[\log n]$ requires $\log(n)$ memory to store the contents of the variables, and at most one alternation per symbol. ∎

Corollary B.2 and Theorem B.4(c) interested us especially because we now have natural classes, Log(*CFL*) and $ASPACE\&Alt[\log(n), \log(n)]$, identified with each of the two intermediate terms in the following containment which is immediate from Theorems A.1 and A.4:

$$NSPACE[\log(n)] \subseteq \mathrm{Var\&Sz}(B\forall)[*, \log(n)] \subseteq \mathrm{Var\&Sz}[*, \log(n)] \subseteq DSPACE[\log^2(n)].$$

The above relations between expressibility and alternating complexity lead to corollaries concerning the relations between expressibility and deterministic complexity. It is, however, interesting to prove the following directly:

THEOREM B.5. *Let* $t(n) \geqslant n$,

$$\bigcup_{k=1,2,\ldots} DSPACE\&TIME[n^k, t(n)^k] = \bigcup_{k=1,2,\ldots} \mathrm{Var\&Sz}[k, t(n)^k].$$

Proof. ($\subseteq$) This is similar to the usual proof that $P \subseteq ASPACE[\log(n)]$. Let M be a deterministic Turing machine running in space n^k and time $t(n)^k$. We describe M's computation via the sentences $\mathrm{Cell}_t(p, a)$ meaning that tape cell number "p" contains symbol number "a" at step t of the computation. Note that the cell location requires $O[\log(n)]$ bits or a constant number of variables to specify. For simplicity we assume a one tape Turing machine. If there were k tapes then a sentence, $\mathrm{Cells}(p_1 \cdots p_k, a_1 \cdots a_k)$ would keep track of all the tape heads in a similar way.

The idea is to say that there exists a triple of cell values a_{-1}, a_0, a_1 in the previous move which lead to a in one move of M, and a_i occurs in cell $p+i$ at time $t=1$. In symbols:

$$\mathrm{Cell}_t(p,a) \equiv (\exists a_{-1} a_0 a_1)\left(a_{-1} a_0 a_1 \to a \;\&\; \bigwedge_{i=-1,0,1} \mathrm{Cell}_{t-1}(p+i, a_i)\right).$$

Here "$a_{-1} a_0 a_1 \to a$" is a finite disjunction over all possible triples, and their consequences. To write our "$\bigwedge_{i=-1,0,1} \mathrm{Cell}_{t-1}(p+i, a_i)$" we use the abbreviation trick:

$$(\forall p')(\forall a')([(p'=p-1 \;\&\; a'=a_{-1}) \text{ or } (p'=p \;\&\; a'=a_0) \text{ or }$$
$$(p'=p+1 \;\&\; a'=a_1)] \Rightarrow \mathrm{Cell}_{t-1}(p', a')).$$

Thus $\mathrm{Cell}_{t(n)^k}(0, q_f)$, meaning that the first cell in M's ID at time $t(n)^k$ is the final state symbol, can be written uniformly with a constant number of variables and $O[t(n)^k]$ symbols.

($\supseteq$): Going the other way we must produce a deterministic Turing machine which given a structure G of size n, and a sentence φ_n with k variables and $t(n)$ symbols, determines if G satisfies φ_n using polynomial space and $t(n)^*$ time.

To test if G satisfies φ_n we examine the parse tree for φ_n. Each of the k variables may take on any of the n values of the universe of G. Thus to each node in the parse tree we can systematically attach the list of at most n^k assignments to the variables which make that node true. The leaves of the parse tree are atomic formulas such as $E(x_2, x_3)$; such a node's associated list contains all those k-tuples $\langle g_1 \cdots g_k\rangle$ such that $G \models E(g_2, g_3)$.

We can pass up the tree towards the root computing the list of k-tuples making each node true, as we go. For example, an "&" node's list is derived by intersecting the two lists it leads to, a "$\forall x_1$" node's list consists of those tuples $\langle g_1 \cdots g_k\rangle$ such that $\langle h, g_2 \cdots g_k\rangle$ appears on the preceding node's list for all values of h.

When we reach the root either our list will contain all n^k possibilities or it will be empty since φ_n has no free variables. G satisfies φ_n if and only if we are in the former case.

Each node's list requires $n^k \log(n)$ space to store. Furthermore at most $\log(t(n))$ lists must be remembered at once—the number of pebbles needed to pebble a tree of size $t(n)$. (Note that in the case in hand φ_n satisfies the syntactic uniformity condition and so is essentially linear. Thus only two lists need be remembered at once.) Thus polynomial space suffices. The time required to compute a node's list from its predecessors is certainly bounded by n^{2k}. Thus the number of steps involved in the entire computation is less than $t(n) \cdot n^{2k}$ which is in turn bounded by $t(n)^*$. ∎

We conclude this section with a corollary which summarizes some of the relationships between classical complexity classes and expressibility with a constant number of quantifiers. Recall that the latter notion comes entirely from logic. The following thus casts the classic problems $P = ?PSPACE$ and $L = ?P$ in a new light. (L is $DSPACE[\log n]$.)

COROLLARY B.6.

(a) $L \subseteq ASPACE\&Alt[\log(n), \log(n)] = \bigcup_{k=1,2,\ldots} \mathrm{Var\&Sz}[k, \log(n)]$.

(b) $P = \bigcup_{k=1,2,\ldots} \mathrm{Var\&Sz}[k, n^k]$.

(c) $PSPACE = \bigcup_{k=1,2,\ldots} \mathrm{Var\&Sz}[k, 2^{n^k}]$.

C. ALTERNATING PEBBLING GAMES

In this section we present a new pebbling game to obtain lower bounds for Var&Sz(w.o. $\leqslant$). This game is a modification of Ehrenfeucht–Fraisse games. (See [10] or [6].) Two players play the p-pebble, m move game on a pair of structures G, H. Player I places pebbles on points from G or H trying to demonstrate a difference between them while Player II matches these points trying to keep the structures looking the same. We will see in Theorem C.1 that if Player II has a win for the p-pebble, m-move game on G and H, then G and H agree on all properties expressible in Var&Sz(w.o. $\leqslant$)$[p, m]$.

DEFINITION. The *p-pebble, m-move* game on G and H is defined as follows: Initially the pebbles, $g_1 \cdots g_p$, $h_1 \cdots h_p$, are off the board. On move i, Player I picks up a pebble g_j (or h_j), $1 \leqslant j \leqslant p$, and places it on a vertex of G (or H). Player II answers by placing h_j (or g_j) on a corresponding point of H (or G). Let $g_j(i)$ be the point on which g_j is sitting just after move i. After each move i, $0 \leqslant i \leqslant m$, define the map f_i as follows:

$$f_i : c^G \to c^H, \qquad g_j(i) \to h_j(i).$$

The map f_i takes the constants in G to the constants in H, and chosen points in G to the respective chosen points in H. We say that Player II *wins* if for each i, $0 \leqslant i \leqslant m$, f_i is an isomorphism of the induced substructures.

The *quantifier rank* of a sentence, φ, is the depth of nesting of quantifiers in φ. Since the quantifier rank of φ is obviously less than or equal to the size of φ, the following theorem shows that the p, m game gives a Var&Sz$[p, m]$ lower bound on the expressibility of any property on which G and H differ.

THEOREM C.1. *Player* II *has a winning strategy for the p, m game on G, H if and only if G and H agree on all sentences with p variables and quantifier rank m.*

We will give the proof, a minor modification of proofs in [10, 5], shortly. First we will give an example. Consider the 4-pebble, $d+1$-move game on undirected graphs G and H where H is disconnected while G is connected with diameter d. See Fig. 1.

Player I wins the game as follows: On the first two moves he puts pebbles h_2, h_3 on vertices a, b such that a and b are in distinct components of H. Player II must place g_2, g_3 on some vertices e, f from G. There is a path of length at most d from e to f. Player I now uses the next $d-1$ moves to walk along this path with pebbles g_0

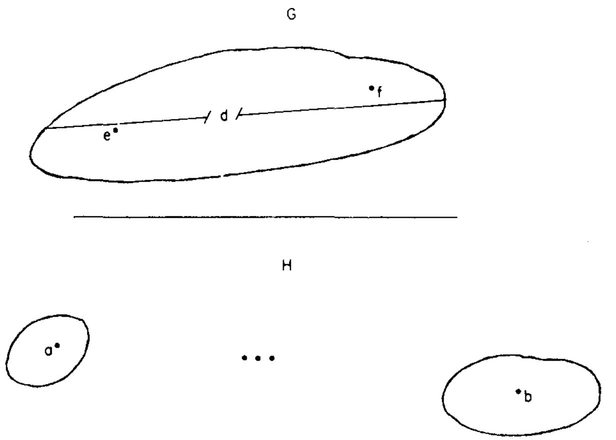

FIG. 1. The 4, $d+1$ game on G and H.

and g_1. Player II must answer with a path in H starting at a, and thus never reaching b. Thus at move $d+1$, two pebbles will coincide in G but not in H and Player I wins.

Notice that Player I's strategy was to follow the following sentence, true in G but not in H: (Let $M(u, v) \equiv E(u, v)$ or $u = v$.)

$$\mathrm{Diam}(d) \equiv \forall x_2 \forall x_3 \exists_3 x_0(M(x_2, x_0) \,\&\, \exists_4 x_1(M(x_0, x_1) \,\&\, \exists_5 x_0(M(x_1, x_0) \,\&\, \cdots \\ \&\, \exists_{d+1} x_i(M(x_{1-i}, x_i) \,\&\, M(x_i, x_3)) \cdots).$$

Also note that there is a sentence equivalent to $\mathrm{Diam}(d)$ with only three variables and $\log(d)+1$ quantifier depth which Player I would have played had he known about it.

Proof of Theorem C.1. We prove a slightly stronger result.

Claim. Let $0 \leqslant k \leqslant p$, and for $1 \leqslant i \leqslant k$, let c_i^G and c_i^H be new constants in G and H respectively. Then the following are equivalent:

(i) Player II has a win for the p-pebble m-move game on G and H when started with the first k pebbles on $c_1^G \cdots c_k^G$ and $c_1^H \cdots c_k^H$, respectively.

(ii) $\langle G, c_1^G \cdots c_k^G \rangle$ and $\langle H, c_1^H \cdots c_k^H \rangle$ agree on all sentences, S, with new constant symbols $c_1 \cdots c_k$, variables $x_1 \cdots x_p$, quantifier rank m, and such that nowhere in S does c_i occur within the scope of a quantifier for x_i.

Note that with $k = 0$ the claim reduces to what we need to show. We prove the claim by induction on m:

Base case. If $m = 0$ then $\langle G, c_1^G \cdots c_k^G \rangle$ and $\langle H, c_1^H \cdots c_k^H \rangle$ agree on all quantifier free sentences if and only if the map from the constants in G (including the new ones) to the respective constants in H is an isomorphism. That is, if and only if Player II has won the 0-move game.

Inductive step: Assume the claim for all $m' < m$.

(i) $\Rightarrow$ (ii). Suppose (i) holds but (ii) does not, and let S be the sentence of quantifier rank m on which $\langle C, c_1^G \cdots c_k^G\rangle$ and $\langle H, c_1^H \cdots c_k^H\rangle$ disagree. If S is of the form $\neg A$, or $A \& B$, then the structures must disagree on one of A or B. Thus we may assume that S is of the form $\exists x_1(M(x_1))$, and $\langle G, c_1^G \cdots c_k^G\rangle \models S$, while $\langle H, c_1^H \cdots c_k^H\rangle \models \neg S$. Player I now places pebble g_1 on some vertex $g_1(1)$ from G so that $\langle G, g_1(1), c_2^G \cdots c_k^G\rangle \models M(c_1)$. Player II must reply by putting h_1 on some $h_1(1)$ such that $\langle H, h_1(1), c_2^H \cdots c_k^H\rangle \models \neg M(c_1)$. Thus Player II still has a winning strategy for the $m-1$ move game, and the two structures differ on $M(c_1)$, a sentence of quantifier rank $m-1$ in which no c_i occurs within the scope of some quantifier for x_i. This violates the inductive assumption.

(ii) $\Rightarrow$ (i). Assume (ii) and let Player I move placing, let us say, pebble, g_1 on $g_1(1)$. Consider the finite collection (up to equivalence) of sentences $S_1(x_1),\ldots, S_r(x_1)$ in the language of G together with variables $x_1 \cdots x_p$, constant symbols $c_2 \cdots c_k$, of quantifier rank $m-1$, such that no c_i occurs within the scope of a (Qx_i) and such that $\langle G, g_1(1), c_2^G \cdots c_k^G\rangle \models S_i(c_1)$.

Let

$$S \equiv (\exists x_1)\left(\bigwedge_{i=1\cdots r} S_i(x_1)\right).$$

Thus

$$\langle G, c_1^G, c_2^G \cdots c_k^G\rangle \models S.$$

Thus, by our assumption, $\langle H, c_1^H \cdots c_k^H\rangle$ also satisfies S. Let $h_1(1)$ be a witness in H for x_1. Now $\langle G, g_1(1), c_2^G \cdots c_k^G\rangle$ and $\langle H, h_1(1), c_2^H \cdots c_k^H\rangle$ agree on all sentences, R, of quantifier rank $m-1$, variables $x_1 \cdots x_p$, and constants $c_1 \cdots c_k$ such that no c_i occurs within the scope of a quantifier for x_i. This is because any such R satisfied by G would be an $S_i(c_1)$ above and therefore also satisfied by H.

Our inductive assumption now shows that Player II wins the remaining $m-1$ moves of the game, proving the claim. This proves Theorem C.1. ∎

Define $G \equiv_{\mathrm{Var}[k]} H$ to mean that G and H agree on all k-variable sentences in the language of their similarity type. What does it mean when G and H agree on all k variable sentences without ordering? Theorem C.1 shows that if Player I chooses any r-tuple of points from G, $r \leqslant k$, then there is a corresponding isomorphic r-tuple from H. Furthermore if Player I adds a point to the tuple in G, and $r < k$, then there is a corresponding points in H which may be added preserving the isomorphism.

We have thus deduced the existence of a relation R on pairs of r-tuples from G and r-tuples from H, i.e., $R \subseteq \bigcup_{r=0,\ldots,k} G^k \times H^k$, satisfying:

(a) $R(\langle\rangle, \langle\rangle)$.

(b) $R(g, h) \Rightarrow g \simeq h$.

(c) $(R(g,h)\ \&\ |g|<k) \Rightarrow (\forall x \in G\ \exists y \in H\ R(\langle g,x\rangle, \langle h,y\rangle))\ \&\ (\forall y \in H\ \exists x \in G\ R(\langle g,x\rangle, \langle h,y\rangle))$.

(d) If $g=\langle g_1 \cdots g_r\rangle$, let $g_i=\langle g_1 \cdots g_{i-1}, g_{i+1} \cdots g_r\rangle$ be the $r-1$ tuple with g_i removed. Then:

$$R(g,h) \Rightarrow R(\hat{g}_i, \hat{h}_i), \qquad i=1,\ldots,r.$$

PROPOSITION C.2. *$G \equiv_{\mathrm{Var}[k]} H$ if and only if there exists a relation R satisfying* (a)–(*d*) *above.*

Proof. It should be clear that R corresponds to Player II's winning strategy in the k-pebble game on G and H. Thus if such an R exists then Player II can always win by matching chosen r-tuples in G with R-related r-tuples in H. Assume $R(\langle g_1(s) \cdots g_k(s)\rangle, \langle h_1(s) \cdots h_k(s)\rangle)$, i.e., the chosen points after move s are R-related. Think of Player I's moving of pebble g_i as two actions. First he picks up g_i. By (d) we know $R(g_i(s), h_i(s)))$. Next he places g_i back on some new point $g_i(s+1)$. By (c) there exists y in H preserving the relation, i.e., with $h_i(s+1)=y$, $R(g(s+1), h(s+1))$. In particular $g(s+1)$ and $h(s+1)$ are isomorphic, and Player II wins.

Conversely, if $G \equiv_{\mathrm{Var}[k]} H$ then define R from Player II's winning strategy as follows:

$$R=\{(\langle x_1 \cdots x_1\rangle, \langle y_1 \cdots y_1\rangle) \mid \text{The } k\text{-pebble game on } G \text{ and } H, \text{ started with } g_i(0)=x_i,\ |g_i(0)=y_i,\ i=1 \cdots r, \text{ is a forced win for Player II.}\}$$

The fact that Player II has a winning strategy for the k-pebble game on G and H gives us (a). Parts (b), (c), and (d) follow from the rules of the game. ▮

D. LOWER BOUNDS FOR Var(w.o. <0)[k]

In this section we will use the alternating pebbling games to prove lower bounds on the number of variables needed to express certain combinatorial properties in the language without $\leqslant$. Recall that the results of section A and B use descriptions of Turing machine computations in first order languages containing $\leqslant$. Thus the results of this section do not translate directly into lower bounds for time and space. Their value is as an intuition and a starting point for similar lower bounds in stronger languages.

Following [8] and [2], we write certain axioms for graphs. First:

$$T_0 \equiv \forall x \forall y(\neg E(x,x)\ \&\ [E(x,y) \Rightarrow E(y,x)]).$$

T_0 says that G is loop free and undirected. We will assume in this section that all graphs satisfy T_0.

Fix k and let $1 \leqslant j \leqslant k-1$. The following sentences, $S_{k,j}$, say that for any choice of distinct vertices, $x_1 \cdots x_j$ and $x_{j+1} \cdots x_{k-1}$, there exists a vertex y different from the x_i's with an edge to every vertex in the first groups and no edge to the second group.

$$S_{k,j} \equiv \forall x_1 \cdots \forall x_{k-1} \left(\left(\bigwedge_{0<i<r<k} x_i \neq x_r \right) \right.$$

$$\left. \Rightarrow \exists y \left[\bigwedge_{0<i<j+1} E(y, x_i) \,\&\, \bigwedge_{j<i<k} (y \neq x_i \,\&\, \neg E(y, x_i)) \right] \right).$$

We use the $S_{k,j}$'s to write T_k, an axiom which says that every conceivable extension of a configuration of $k-1$ points to a configuration of k points is realizable.

$$T_k \equiv \bigwedge_{0<j<k} S_{kj}.$$

A counting argument shows that almost all graphs satisfy T_k. Define $P_n(S)$, the probability that a graph of size n satisfies a sentence S, as follows:

$$P_n(S) \equiv \#\{G \mid G \models S, |G| = n\} / \#\{G \mid |G| = n\}.$$

THEOREM D.1 [8, 2]. *For any fixed $k > 0$, $\lim_{n\to\infty}[P_n(T_k)] = 1$.*

Proof. Given $j < k$, and distinct vertices $x_1 \cdots x_{k-1}$ what is the probability that a random vertex y is a witness for $S_{k,j}$? It's just the probability that the $k-1$ possible edges $E(x_i, y)$ are correctly present or absent, i.e., $1/2^{k-1}$.

Thus the probability that none of a random $n-(k-1)$ vertices is a witness for $S_{k,j}$ is:

$$\alpha^{n-k+1}, \qquad \text{where} \quad \alpha = 1 - (1/2^{k-1}).$$

The probability that any of the fewer than n^k sequences, $x_1 \cdots x_{k-1}, j$, cause T_k to fail is less than

$$n^k \cdot \alpha^{n-k+1}$$

and this last probability goes to 0 as n goes to infinity. ∎

We are interested in T_k because of the next result:

THEOREM D.2. *For any two graphs G and H,*

$$(G \models T_k \,\&\, H \models T_k) \Rightarrow G \equiv_{\mathrm{Var}[k]} H.$$

Proof. T_k says that every $k-1$ tuple may be extended to a k tuple in any conceivable way. It follows that the relation:

$$R = \{(\langle a_1 \cdots a_r\rangle, \langle b_1 \cdots b_r\rangle) \mid 0 \leqslant r \leqslant k, a_i \in G, b_i \in H, \,\&\, \langle a_1 \cdots a_r\rangle \simeq \langle b_1 \cdots b_r\rangle\}$$

satisfies (a)–(d) of Proposition C.2. Therefore $G \equiv_{\mathrm{Var}[k]} H$. ∎

COROLLARY D.3. *Graph Isomorphism is not in* Var(w.o. $\leqslant$)$[k]$.

Proof. If GraphIso where in Var(w.o. $\leqslant$)$[k]$ then there would be sentences F_1, F_2... with k variables each such that for graphs G and H of size n,

$$\langle G, H\rangle \models F_n \leftrightarrow G \simeq H.$$

Here $\langle G, H\rangle$ is the structure consisting of a disjoint union of G and H with a monadic predicate true for exactly the points of G. By Theorem D.1 there exist two non-isomorphic graphs G_k and H_k both satisfying T_k. Clearly $\langle G_k, G_k\rangle \models F_n$. But by Theorem D.2, $G_k \equiv_{\mathrm{Var}[k]} H_k$. It follows that Player II wins the k-pebble game on $\langle G_k, H_k\rangle$ and $\langle G_k, G_k\rangle$. Her strategy is to answer points in the first component with the same point in the other copy of G_k, and to use Player II's winning strategy for the k-pebble game on G_k and H_k to answer moves in the second component. This strategy preserves an induced isomorphism between points chosen in each component and is thus a win for Player II. It follows that $\langle G_k, G_k\rangle \equiv_{\mathrm{Var}[k]} \langle G_k, H_k\rangle$. Thus,

$$\langle G_k, H_k\rangle \models F_n, \text{ but } G_k \text{ is not isomorphic to } H_k.$$

This contradiction proves the corollary. ∎

Almost all graphs have a Hamilton circuit; however, in [8] it is shown that for any k there is a graph H_k which satisfies T_k and yet has no Hamilton circuit. It follows that there exist two graphs, G_k, H_k, both satisfying T_k and yet differing on the property of having a Hamilton circuit. Thus:

THEOREM D.4. *"Hamilton Circuit" is not in* Var(w.o. $\leqslant$)$[*]$.

Using similar techniques we can show the following:

THEOREM D.5. Clique($k+1$) *is not in* Var(w.o. $\leqslant$)$[k]$.

Proof. Recall that Clique($k+1$) is the set of graphs with a complete subgraph of size $k+1$. Clearly any graph satisfying T_{k+1} is in Clique($k+1$). We show that there exists a graph $H_k \models T_k$ such that H_k has no $k+1$ clique. Define the graph $A_n = (V_n, E_n)$ as follows:

$$V_n = \{\langle i, j\rangle \mid 1 \leqslant i \leqslant k, 1 \leqslant j \leqslant n\},$$
$$E_n = \{(\langle i_1, j_1\rangle, \langle i_2, j_2\rangle) \mid i_1 \neq i_2\}.$$

Notice that A_n has no $k+1$ clique because any set of $k+1$ vertices will have two with the same first coordinate.

Let $A'_n = (V_n, E'_n)$ be a random subgraph of A_n, i.e. each edge of E_n has probability 1/2 of being in E'_n. Now $\lim_{n\to\infty} \mathrm{Prob}(A'_n \models T_k) = 1$. (This follows from the same argument as in the proof of Theorem D.1, noting that every $k-1$ tuple from V_n has n points potentially satisfying T_k.) Let H_n be such a random A'_n. Thus H_n satisfies T_k but has no $k+1$ clique. ∎

F. CONCLUSIONS

We feel that first order expressibility is a natural way to obtain both upper and lower bounds. The alternating pebbling games make the finding of optimal descriptions of graph properties (without ordering) a tractable problem. Furthermore our simulation theorems show that optimal sentences (with ordering) for a property C can be easily translated to nearly optimal algorithms for checking C.

The following general areas of exploration are sugested:

(1) Find upper and lower bounds on Var&Sz(w.o. $\leqslant$) for a collection of graph problems such as planarity, graph homeomorphism, vertex matching, etc.

(2) Improve the simulations of Section B, and then try to prove optimality. Exactly how many variables are needed to describe a $DTIME[n^k]$ computation?

(3) Develop techniques to prove lower bounds on Var&Sz, i.e., with ordering. This seems worthwhile but hard. One possible method would be to consider sentences true for "most" orderings. See [17, 4] for some results concerning the probability that a formula is satisfied by a large finite structure. Other possible techniques are discussed in [13, 15].

ACKNOWLEDGMENTS

Warm thanks to Juris Hartmanis, my thesis adviser. Many thanks to John Hopcroft, Albert Meyer, and Michael Morley for helpful technical discussions. Thanks to MIT's Laboratory for Computer Science for letting me visit since June, 1980. Much thanks to the referees whose patient and thorough readings of this paper have helped to remove many of the ambiguities and errors.

REFERENCES

1. A. AHO, J. HOPCROFT, AND J. ULLMAN, "The Design and Analysis of Computer Algorithms," Addison–Wesley, Reading, Mass., 1974.
2. A. BLASS AND F. HARARY, Properties of almost all graphs and complexes, *J. Graph Theory* **3** (1979), 225–240.
3. S. CHANDRA AND L. STOCKMEYER, Alternation, "*Proc. 17th FOCS*, 1976,"pp. 98–108.
4. K. COMPTON, Ph.D. Thesis, University of Wisconsin, Madison, 1980.
5. A. EHRENFEUCHT, An application of games to the completeness problem for formalized theories, *Fund. Math.* **49** (1961), 129–141.
6. H. ENDERTON, "A Mathematical Introduction to Logic," Academic Press, New York, 1972.
7. R. FAGIN, "Generalized first-order spectra and polynomial-time recognizable sets, *in* "Complexity of Computation," (R. Karp, Ed.), SIAM-AMS Proc. No. 7, pp. 43–73, Amer. Math. Soc., Providence, R. I., 1974.
8. R. FAGIN, Probabilities on finite models, *J. Symbol Logic* **41**, No. 1 (1976), 50–58.
9. M. FISCHER, AND M. RABIN, Super-exponential complexity of Presburger arithmetic, *in* "Complexity of Computation" (R. Karp, Ed.), SIAM-AMS Proc. No. 7, pp. 27–41, Amer. Math. Soc., Providence, R.I., 1974.
10. R. FRAISSE, Sur les classifications des systems de relations, *Publ. Sci. Univ. Alger* I (1954).

11. J. HARTMANIS, N. IMMERMAN, AND S. MAHANEY, One-way Log tape reductions, *in* "*Proc. 19th FOCS*, 1978," pp. 65–72.
12. N. IMMERMAN, Length of predicate calculus formulas as a new complexity measure, *in* "*Proc. 20th FOCS*, 1979," pp. 33747.
13. N. IMMERMAN, "First Order Expressibility as a New Complexity Measure," Ph.D. Theis, Cornel University, August, 1980.
14. N. IMMERMAN, Upper and lower bounds for first order expressibility, *in* "Proc. 21st FOCS, 1980," pp. 74–82.
15. N. IMMERMAN, Number of quantifiers is better than number of tape cells, *J. Comput. System Sci.*, in press.
16. D. KOZEN, On parallelism in Turing machines, *in* "*Proc. 17th FOCS*, 1976," pp. 89–97.
17. J. LYNCH, Almost sure theories, *Ann. Math. Logic* **18** (1980, 91–135.
18. J. REIF, Universal games of incomplete information, *in* "*Proc. 11th SIGACT*, 1979," pp. 288–308.
19. W. RUZZO, Tree-size bounded alternation, *in* "*Proc. 11th SIGACT*, 1979," pp. 352–359.
20. W. RUZZO, On uniform circuit complexity, *in* "*Proc. 20th FOCS*, 1979," pp. 312–318.
21. W. SAVITCH, Maze recognizing automata and nondeterministic tape complexity, *J. Comput. System Sci.* **7** (1973), 389–403.
22. L. SUDBOROUGH, On the tape complexity of deterministic CFL's *J. Assoc. Comput. Mach.* No. 3 (1978), 405–414.