Upper and Lower Bounds for First Order Expressibility

Neil Immerman

Computer Science Department
Cornell University
Ithaca, New York 14853

Abstract

We continue the study of first order expressibility as a measure of complexity, introducing the new class Var &Sz[v(n),z(n)] of languages expressible with v(n) variables in sentences of size z(n). We show that when the variables are restricted to boolean values:

$$\text{BVar \&Sz}[v(n),z(n)] = \text{ASPACE \&TIME}[v(n),t(n)]$$

That is variables and size correspond precisely to alternating space and time respectively. Returning to variables ranging over an n element universe, it follows that:

$$\text{Var}[O(1)] = \text{ASPACE}[\log n] = \text{PTIME}$$

That is the family of properties uniformly expressible with a constant number of variables is just PTIME.

These results hold for languages with an ordering on the objects in question, e.g. for graphs a successor relation on the vertices. We introduce an "alternating pebbling game" to prove lower bounds on the number of variables and size needed to express properties without successor. We show, for example, that k variables are needed to express Clique(k), suggesting that this problem requires DTIME[n^k] .

Introduction and Summary

In [Imm79] 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:

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

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

Reading some papers by Ruzzo ([Ruz79a], [Ruz79b]), on simultaneous resource bounds, we tried to find analagous results for first order expressibility. First we reexamined our proof of the above containment for f(n) =log(n), i.e:

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

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 BUVar &Sz[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:

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

Savitch's simulation of NSPACE[logn] by DSPACE[$\log^2$(n)] may be optimal, but one way of thinking of the difference between the classes is that DSPACE[$\log^2$(n)] can simulate log(n) universal quantifiers ranging from 1 to n. We conjecture that all three containments in the above chain are proper, but none are known to be.

It turns out that BUVar&Sz[*,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, 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. Define Var[*] = $\bigcup_{k=1,2...}$ Var &Sz[k,n^k] -- those properties expressible with a constant number of variables. It turns out that Var[*] is identical to polynomial time! The identity between P and Var[*] is a very pleasing result both because it indicates that first order expressibility is a fruitful view of complexity, and because it is another demonstration of the fundamental importance and model independent nature of P.

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. Our feeling at the time 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 n^{th} sentence of a Var &Sz[k,z(n)] property is just z(n) repetitions of a fixed block of k quantifiers. This new definition of uniformity makes Var[*] entirely a notion from logic and thus increases the interest of the fact that it is equal to P.

* Research partly supported by NSF grant no. MCS 78-00418

CH1498-5/80/0000-0074$00.75 © 1980 IEEE

Now that we know that DTIME[n^k] is closely related to Var[k] 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 [Ehr61] or [Fra54]), 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 successor relation, Suc(-,-), 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 [Imm79] that Suc(-,-) 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 successor 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 Suc. (Of course k variables suffice -- just say there exist $x_1 \ldots x_k$ forming a clique.) This is a plausability argument that Clique(k) is not in Var[k-1] . If we could prove the latter result, i.e. that Clique(k) cannot be expressed with k-1 variables in the language with Suc, then it would follow that the general clique problem is not in Var[*]. From this it would follow that P≠NP.

In Section D we also consider the graph isomorphism problem. If we knew, for example, that GraphIso were in DTIME[$n^{\log(n)}$] then it would follow that this property could be written with log(n) variables without successor. Thus a pair of graphs, <G,H>, satisfy F_n , a sentence with log(n) variables, if and only if they are isomorphic. Clearly $\langle G,G\rangle \models F_n$. Now suppose that $G \equiv_{Var[\log n]} H$, i.e. G and H agree on all log(n) variable sentences. It follows that $\langle G,H\rangle \models F_n$, and thus G and H are isomorphic. We have shown:

$$\text{GraphIso} \in \text{Var(w.o. Suc)}[v(n)] \leftrightarrow$$
$$\forall G \forall H \left(|G|=|H|=n \Rightarrow [G \simeq H \leftrightarrow G \equiv_{Var[v(n)]} H] \right)$$

In DTIME[n^{4v}] we can check if $G \equiv_{Var[v]} H$. Thus a near optimal algorithm for GraphIso may be to find the correct v(n) in the above equivalence, and check if $G \equiv_{Var[v(n)]} H$.

We mentioned above a proof that Clique(k) cannot be expressed in k-1 variables without Suc. This is shown by building graphs G and H such that $G \equiv_{Var[k-1]} H$ and yet G has a k-clique while H does not. Clearly G and H are not isomorphic. Thus we have a k variable (without Suc) lower bound on GraphIso. This is a plausability argument that GraphIso is not in P.

In the following pages we give: (A) Definitions and motivations; (B) Statements of some of the main relationships between expressibility and Turing machine Time and Space; (C) The alternating pebbling game; (D) Probabilistic graph arguments following [Fag76] and [BlHa79] showing that Hamilton Circuit, Clique, and GraphIso are not in Var(w.o. Suc)[*]; and (E) Conclusions and directions for future research.

Section A: Definitions and Motivations

We propose to study the complexity of a condition, C, by asking, "How difficult is it 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, the vertices, together with a binary edge relation E(-,-) on V. This is a logical structure of similarity type $\tau_G = \{E(-,-)\}$. The language of a type, τ, $L[\tau]$, consists of the sentences built up from the symbols of τ using the logical connectives &, "or", ¬, ⇒, variables x,y, ... , =, and quantifiers, ∃x and ∀x, ranging over the universe. For example, consider the following sentence from $L[\tau_G]$:

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

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:

$$\text{GAP} \quad = \quad \{\; G \mid a \rightarrow^* \rightarrow b \;\}$$

GAP is known to be complete for NSPACE[log n]. (See [Sav73].) We show in [Im80a] 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:

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

$$P_n(x,y) \equiv \exists z \left(P_{n/2}(x,z) \;\&\; P_{n/2}(z,y) \right) \qquad (2)$$

Equation (2) defines P_n in a way that increases the quantifier depth 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. [FiRa74]). The trick uses universal quantifiers to permit us to write $P_{n/2}$ only once on the right. Thus:

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

We have now written P_n with O[log n] symbols, thus proving that GAP is in 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:

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

We have succeeded in expressing GAP by a uniform sequence of sentences, $\{P_n(a,b) \mid n \geq 1\}$, such that P_n has 5 variables and size O[logn]. 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 ... from $L[\tau \cup \{Suc\}]$ such that :

a. For all structures G of type τ with $|G| \leq n$, and for all $Suc_0(\cdot,\cdot)$ a valid successor relation on the universe of G,

$$G \in C \leftrightarrow \langle G, Suc_0 \rangle \models F_n$$

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

As Ruzzo has shown in [Ru79b], uniformity conditions may be greatly varied without significantly changing a definition. The following condition will suffice in what follows:

Uniformity Condition (*): The map $n \rightarrow F_n$ is generable in DSPACE[v(n)•logn] 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:

$(\exists x \,.\, M)[\,...] \equiv \exists z\,[M \,\&\, ...]$ read, "There exists x such that M."
$(\forall x \,.\, M)[\,...] \equiv \forall x[M \Rightarrow ...]$ read, "For all x such that M."

Now we can write Equation (4) more compactly as:

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

Here $M_3 = [u=x \,\&\, v=z$ or $u=z \,\&\, v=y]$, and $M_5 = [x=u \,\&\, y=v]$. Equation (5) gives a model for the following totally syntactical definition of uniformity for Var&Sz[v,z(n)] :

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

$$F_n \equiv A \left((Q_1 x_1 \,.\, M_1) \ldots (Q_v x_v \,.\, M_v) \right)^{c \cdot z(n) \text{ times}} B$$

We adopt (**) as our definition of uniformity for Var &Sz[v,z(n)] when v is a constant, otherwise we use (*). It follows that GAP is in Var &Sz[5,logn], NSPACE[logn] is in Var &Sz[k,logn], and indeed:

Theorem A.1: For $s(n) \geq \log(n)$,

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

proof sketch: The proof is nearly the same as for Theorem 2 in [Imm79]. We showed there that $NSPACE[s(n)] \subseteq Size[s(n)^2/\log(n)] \subseteq DSPACE[s(n)^2]$. 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 equation (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 Theorem A.1 we use equation (4) instead. Thus only a constant number of ID's, requiring O[s(n)/log(n)] variables, must be remembered at once. ∎

Let's return to Equation (4) and notice that in simulating an NSPACE[logn] 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 Equation (4) by "$\forall b$", where b is boolean valued. Thus:

$$P_n(x,y) \equiv \exists z \forall b \exists u \exists v \left([(b=0 \,\&\, u=x \,\&\, v=z) \text{ or } (b=1 \,\&\, u=z \,\&\, v=y)] \;\&\; \exists x \exists y\, [x=u \,\&\, y=v \;\&\; P_{n/2}(x,y)\,] \right) \qquad (6)$$

Define BUVar&Sz[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. Thus it is easy to see that GAP is in BUVar &Sz[k,logn], and more generally,

Theorem A.2: For $s(n) \geq \log(n)$,

$$NSPACE[s(n)] \subseteq BUVar\,\&Sz[O[s(n)/\log(n)], s(n)^2/\log(n)]$$

Section B: Variables & Size versus Time & Space

Recall a definition and result of Sudborough [Sud78]:

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

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

In [Ruz79a] Ruzzo 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 all 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 auxilliary pda's via,

Fact (Ruzzo): $ASPACE\,\&TS[s(n),z(n)^*] = AuxPDA[s(n),z(n)^*]$.

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 existential.

Theorem B.1: $BUVar\&Sz[O[s(n)/\log(n)], z(n)] = ASPACE\&TS[v(n)\log(n), *^{z(n)}]$

proof: ($\subseteq$): Given an input structure G with n element universe we can generate F_n , the n^{th} 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 $x_1 \ldots x_{v(n)}$ in our v(n)log(n) memory. Note that each non-boolean 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.

($\supseteq$): Here we follow a proof of Ruzzo [Ruz79a]. 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. Thus,

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

Here $\text{Accept}(r,\langle q_1 \ldots q_k\rangle,z)$ 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 $q_1 \ldots q_k$.

Our only trouble is to insure that the list $\langle q_1 \ldots q_k\rangle$ 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,

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

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,

$$\exists p \, (\exists s_1 s_2 s_3 \text{, a permutation of } q_1 q_2 q_3)\left(\text{Accept}(r,\langle s_1 ,p\rangle,z) \ \& \ \text{Accept}(p,\langle s_2 ,s_3 \rangle,z) \right)$$

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

Corollary B.2:

a. $\text{BUVar\&Sz}[*,\log(n)] = \text{Log(CFL)}$

b. $\text{Var}[O[v(n)]] = \text{DTIME}[n^{O[v(n)]}]$

c. $\text{Var}[*] = \text{PTIME}$

proof:

(a): From the above theorem, together with the results from Ruzzo and from Sudborough:

$$\begin{aligned} \text{BUVar \&Sz}[*,\log(n)] &= \text{ASPACE\&TS}[\log(n),n^*] \\ &= \text{AuxPDA}[\log(n),n^*] \\ &= \text{Log(CFL)} \end{aligned}$$

(b): By our uniformity condition the n^{th} sentence of a Var[v(n)] property can be generated in DSPACE[v(n)log(n)] and so it is of size at most $n^{O[v(n)]}$ Restricting the universal quantifiers to be boolean at worst increases the size by a factor of log(n). Thus:

$$\begin{aligned} \text{Var}[O[v(n)]] &= \text{Var\&Sz}[O[v(n)],n^{O[v(n)]}] \\ &= \text{BUVar\&Sz}[O[v(n)],n^{O[v(n)]}] \end{aligned}$$

Using Theorem B.1,

$$\begin{aligned} &= \text{ASPACE\&TS}[v(n)\log(n),2^{n^{O[v(n)]}}] \\ &= \text{ASPACE}[v(n)\log(n)] \\ &= \text{DTIME}[n^{O[v(n)]}] \end{aligned}$$

(c): This is a special case of (b). ∎

The above corollary rounds out a pleasing relationship between expressibility and computation. We have shown:

1. The size of a sentence needed to express condition C is polynomially related to the amount of memory space needed to check if C holds for an input, and,
2. Conditions which can be expressed with v variables are just those conditions which can be checked in $\text{DTIME}[n^{O[v]}]$

The next theorem generalizes the above results giving a remarkably close relationship between expressibility and alternating machine complexity. Let BVar &Sz[v(n),z(n)] be those properties expressible in a sentence with v(n) *boolean* variables and size z(n). For each predicate symbol, E, we add the predicate symbols, E_1 , E_2 ... where E_n $(b_1 \ldots b_{\log n}$, $c_1 \ldots c_{\log n})$ means E(b,c) where b has binary vertex number $b_1 \ldots b_{\log n}$.

Theorem B.3: $\text{ASPACE\&TIME}[s(n),t(n)] = \text{BVar\&Sz}[O[s(n)],t(n)]$

proof: ($\supseteq$): Given an input structure G with n element universe we can generate F_n the n^{th} element in our uniform sequence. We must show that in ASPACE&TIME[s(n),t(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 the variables $b_1 \ldots b_{s(n)}$ in our memory. At quantifiers $\exists b_i$ or $\forall b_i$ we make the appropriate existential or universal choice of a new value for b_i . Similarly at &'s (or "or"s) we universally (or existentially) choose one branch and proceed. We have G and the values of the variables so we may check the truth of atomic formulas: $b = 0$, or $E(b_1 \ldots c_{\log n})$, in number of steps a constant times their length.

($\subseteq$): Going the other way we must write the sentence, $\text{Accept}_t(r)$ meaning that alternating Turing machine M when started at instantaneous description r will reach acceptence in t steps. We accomplish this by saying that if r is existential then there exists some next step x and $\text{Accept}_{t-1}(x)$, whereas if r is universal then for all next steps x, $\text{Accept}_{t-1}(x)$.

A technical difficulty here is that if at each step we recopy the entire ID then the size of the resulting sentence will be s(n)t(n). To simplify the problem let us assume that the moves of our Turing machine alternate at every step and branch into at most two moves. Thus we can write,

$$\text{Accept}[ID_0] = \exists b_1 \forall b_2 \ldots Q_s b_s \exists ID_1 \left(ID_0 \rightarrow b_1 \ldots b_s \rightarrow ID_1 \ \& \ \text{Accept}[ID_1] \right)$$

Here "$ID_0 \rightarrow b_1 \ldots b_s \rightarrow ID_1$" means that there is a computation whose i^{th} move makes the b_i^{th} choice and leads from ID_0 to ID_1 in s steps. This is a deterministic computation of length s and, although we omit the details, it can be asserted to exist with O[s] symbols. ∎

We have demonstrated an exact relationship between alternating Turing machines and quantified boolean formulas. If we return to the more natural language of the input structures, i.e. variables ranging from 1 to n, then the needed number of variables to simulate s(n) space becomes s(n)/log(n). It is not clear, however, how to do better than size t(n) to simulate time t(n) because the machine might go through t(n) alternations. Thus we can only show:

Corollary B.4: For $s(n) \geq \log(n)$,

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

Let ASPACE&Alt[s(n),a(n)] be those problems accepted in alternating space s(n) with at most a(n) alternations between existential and universal states. Then:

Corollary B.5: Let $s(n) \geq a(n)\log(n)$. Then:

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

and in particular, $\text{ASPACE\&Alt}[\log n, \log n] = \text{Var\&Sz}[*,\log n]$

proof: To assert the acceptence of an ASPACE&Alt[s(n),a(n)] computation we assert the existence of a(n) ID's where the alternations occur. Each path between the ID's has no alternations and so can be expressed in Var&Sz[O[s(n)/log(n)] , $s(n)^2/\log(n)$] by Theorem B.1 . We write $\text{ACCEPT}_a(ID_0)$ to mean that ID_0 leads to acceptence in a alternations:

$$\text{ACCEPT}_{2a}(ID_0) \equiv \exists ID_1 \left(\text{EPath}(ID_0, ID_1) \ \& \ \text{ACCEPT}_{2a-1}(ID_1) \right)$$

$$\text{ACCEPT}_{2a-1}(ID_1) \equiv \forall ID_0 \left(\text{APath}(ID_1, ID_0) \Rightarrow \text{ACCEPT}_{2(a-1)}(ID_0) \right)$$

Note that in the above EPath(x,y) and APath(x,y) are the Var&Sz[O[s(n)/log(n)],$s(n)^2$/log(n)] formulas which assert the existence of a computation path from x to y all of whose intermediate states are existential, respectively universal. We can use the abbreviation trick to conglomerate the two terms on the right, making the size of Accept_a (-) equal to:

$$a \cdot s(n).\log(n) + \text{Size(EPath)} = (a(n) + s(n))s(n)/\log(n), \text{ as desired.}$$ ∎

The above corollary interested us especially because we now have natural classes, Log(CFL) and ASPACE&Alt[logn,logn], identified with both of the the intermediate terms in the following containment:

Corollary B.6: $\text{NSPACE}[\log n] \subseteq \text{BUVar\&Sz}[k,\log n] \subseteq \text{Var\&Sz}[k,\log n] \subseteq \text{DSPACE}[\log^2(n)]$

Corollary B.6 which comes immediately from Theorems B.1 and B.2 sheds some light on the difference between $\text{DSPACE}[\log^2(n)]$ and NSPACE[logn] . We conjecture that all three containments above are proper, but it is not even known that NSPACE[log(n)] $\neq$ $\text{DSPACE}[\log^2(n)]$.

It makes sense to consider a sentence with k variables which is of length greater than n^k . Similarly we can consider an ASPACE[logn] machine which runs for more than n^k steps. Generalize the definition of ASPACE &TIME[s(n),t(n)] to be the family of languages accepted by an ASPACE[s(n)] machine with a t(n) clock. Such a machine's accepting configuration is an accept state with the clock equal to 0, however the machine is never allowed to look at its clock. With this definition Theorem B.3 makes sense and is true for all t(n). It is now possible to ask, "What is ASPACE &TIME[s(n),t(n)] with $t(n) > 2^{s(n)}$?"

Corollary B.7:

a. $\text{ASPACE\&TIME}[\log(n),n^*] = \text{Var\&Sz}[*,n^*] = \text{PTIME}$

b. $\text{ASPACE\&TIME}[\log(n),2^{n^*}] = \text{Var\&Sz}[*,2^{n^*}] = \text{PSPACE}$

proof: We have already shown line (a) in Corollary B.2 . The left hand equality of line (b) follows from Corollary B.4. The two containments of the second equality are proved as follows:

(Var &Sz[*,2^{n^*}] ⊂ PSPACE): We are given an input structure, G, of size n, and a sentence, F_n ,with k variables and size 2^{n^k}. Check whether $G \models F_n$ as follows:

Assume that all ¬'s have been pushed through to the inside and consider the parse tree for F_n. Each of the k variables may take on any of the n values of the universe of G. Starting at the leaves of the parse tree make a list of all the k-tuples of assignments which make the given nodes true in G. We can pass up the tree toward the root computing the values making each node true as we go.

For example, an "&" node's list is gotten by intersecting the two lists it leads to, a "$\forall x_1$ " node gets those tuples $\langle -, x_2 \ldots x_k \rangle$ which are in the preceding node's list with all values of x_1 .

When we reach the root either our list will have all n^k possibilities or it will be empty, since F_n has no free variables. $G \models F_n$ if and only if we are in the former case. At most two of the k-tuples must be rembered at once, so PSPACE suffices.

(Var &Sz[*,2^{n^*}] ⊃ PSPACE): We show how to describe a computation of Turing machine M running in DSPACE&TIME[s(n),t(n)]. We will construct formulas, $C_t(p,x)$, meaning, "Symbol x occurs in cell p at time t." $C_t(p,x)$ will be written with O[log(s(n))/log(n)] variables.

The idea is to say that there exists a triple of cell values x_{-1} x_0 x_1 in the previous move which lead to x in one move of M, and x_i occurs in cell p+i at time t-1. In symbols:

$$C_t(p,x) \equiv \exists x_{-1}x_0x_1 \left(x_{-1}x_0x_1 \rightarrow 1 \rightarrow x \ \& \ \bigwedge_{i=-1 \ldots 1} C_{t-1}(p+i,x_i) \right)$$

We can use the abbreviation trick to write C_{t-1} only once on the right. Note that p is a O[log(s(n))/log(n)]-tuple of variables coding

a tape location less than s(n). The sentence C_t can be written with O[log(s(n))/log(n)]- variables and size O[t(n)]. For a polynomial s(n) this gives a constant number of variables as desired. ∎

Right now line (a) is what we call ASPACE[log(n)] and Var[*], but it is not clear whether or not line (b) deserves at least the latter name.

Section C: Alternating Pebbling Games

In this section we present a new pebbling game to obtain lower bounds for Var&Sz(w.o. Suc). This game is a modification of Ehrenfeucht-Fraisse games. (See [Fra54] or [Ehr61].) 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. Suc)[p,m].

Definition: The p pebble, m move game on G and H is defined as follows: Initially the pebbles, $g_1 \ldots g_p$, $h_1 \ldots h_p$, are off the board. On move i, Player I picks up a pebble g_j (or h_j), $1 \le j \le p$, 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, $0 \le i \le m$, define the map f_i as follows:

$$f_i : c^G \rightarrow c^H , \quad g_j (i) \rightarrow h_j (i)$$

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, $0 \le i \le m$, f_i is an isomorphism of the induced substructures.

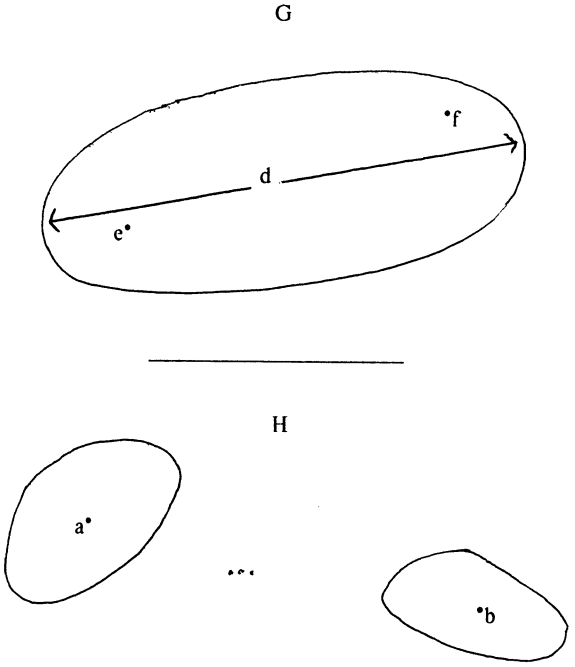
The quantifier rank of a sentence, φ, is the depth of nesting of quantifiers in φ. Since the quantifier rank of φ is obviously less than the size of φ, the following theorem shows that the p,m game gives a Var&Sz[p,m] lower bound on the expressibility of any property on which G and H differ.

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 [Fra54] and [Ehr61], 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.

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 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.

Figure 1: The 4 , d+1 game on G and H.



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

$$\text{Diam}(d) \equiv \forall x_2 \forall x_3 \exists_3 x_0 \left(M(x_2, x_0) \;\&\; \exists_4 x_1 \left(M(x_0, x_1) \;\&\; \exists_5 x_0 \left(M(x_1, x_0) \;\&\ldots\& \exists_{d+1} x_i \left(M(x_{1-i}, x_i) \;\&\; M(x_i, x_3) \right) \right) \ldots \right) \right)$$

Also note that there is a sentence equivalent to Diam(d) with only 3 variables and log(d)+1 quantifier depth which Player I would have played had he known about it.

proof of Theorem C.1: (⇒): Suppose there is a sentence S with p variables and quantifier rank m such that G satisfies S but H does not. We must show that Player I wins the p pebble, m move game on G and H. This is proved by induction on m:

base case: If m=0 then G and H differ on a quantifier free sentence, i.e. the constants in G satisfy a formula that the constants in H do not. Thus they are not isomorphic so Player I wins the 0 move game.

inductive step: If S is of the form ¬A, or A&B, then G and H must disagree on one of A or B. Thus we may assume that S is of the form $\exists x_1 M(x_1)$. Here Player I places pebble g_1 on some vertex g_1 (1) from G so that $G \models M(g_1(1))$. No matter what II answers we will have $H \models \neg M(h_1(1))$. Thus by induction Player I will win.

Note that in the inductive step we have placed pebble g_1 so we must consider what happens if we later need g_1 again. The answer is that in $M(g_1(1))$ the substitution of $g_1(1)$ is made for all *free* occurrences of x_1. If later on in the game we need to place g_1 again it will be for some sentence $S' = Qx_1\, N(x_1)$. Inside S' all occurrence of x_1 are bound by Qx_1, thus $g_1(1)$ does not occur and pebble g_1 may be safely reused.

($\Leftarrow$): Conversely suppose that G and H agree on all sentences with p variables and quantifier rank m. We must show that Player II wins the p pebble, m move game on G and H. To facilitate an inductive proof we must slightly strengthen our claim. We prove the following:

Claim: Let $k \leq p$ and supose that $\langle G, c_1^G \ldots c_k^G\rangle$ and $\langle H, c_1^H \ldots c_k^H\rangle$ agree on all sentences S with new constants $c_1 \ldots c_k$, variables $x_1 \ldots x_p$, quantifier rank m, and such that nowhere in S does c_i occur within the scope of a quantifier for x_i. Then Player II has a win for the p pebble, m move game on G and H when started with the first k pebbles on $c_1^G \ldots c_k^G$ and $c_1^H \ldots c_k^H$ respectively.

Note that with $k=0$ the claim reduces to what we need to show.

We prove the claim by induction on m. For $m=0$ the map from the constants in G to the constants in H and $c_1^G \ldots c_k^G$ to $c_1^H \ldots c_k^H$ must be an isomorphism or else there would be a quantifier free sentence on which the two structures disagree.

For the inductive step asume that the premise of the claim holds and let Player I move placing, let us say, pebble g_1 on $g_1(1)$.

Consider the (finite) collection of sentences $S_1(x_1) \ldots S_r(x_1)$, in the language of G together with constants $c_2 \ldots c_k$ such that $\langle G, c_2^G \ldots c_k^G\rangle \models S_i(g_1(1))$. Let $S \equiv \exists x_1 \left(\wedge_{i=1 \ldots r} S_i(x_1) \right)$. Thus, $\langle G, c_2^G \ldots c_k^G\rangle \models S$.

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

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

Before we apply Theorem C.1 it is useful to give a slightly different characterization of $G \equiv_{Var[k]} H$. What does it mean when G and H agree on all k variable sentences without successor? The idea is that if Player I chooses any r-tuple of points from G, $r \leq k$, 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 point 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. $R \subseteq \cup_{r=0 \ldots k} G^k \times H^k$, satisfying:

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

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

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

d. If $g = \langle g_1 \ldots g_r\rangle$, let $\hat{g}_i = \langle g_1 \ldots g_{i-1}, g_{i+1} \ldots g_r\rangle$ 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: $G \equiv_{Var[k]} H$ 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 $R(\langle g_1(s) \ldots g_k(s)\rangle, \langle h_1(s) \ldots h_k(s)\rangle)$, 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 $R(\hat{g}_i(s), \hat{h}_i(s))$. 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 $G \equiv_{Var[k]} H$ then define R from Player II's winning strategy as follows:

$$R = \left\{ (\langle x_1 \ldots x_r\rangle, \langle y_1 \ldots y_r\rangle) \;\middle|\; \begin{array}{l}\text{The k pebble game on G and H, started} \\ \text{with } g_i(0)=x_i,\ h_i(0)=y_i \quad i=1 \ldots r, \text{ is a} \\ \text{forced win for Player II.}\end{array} \right\}$$

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

Section D: Lower Bounds for Var(w.o. Suc)[k]

Following [Fag76] and [BlHa79], we write certain axioms for graphs. First:

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

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

$$S_{k,j} \equiv \forall x_1 \ldots \forall x_{k-1} \left(\left(\wedge_{0<i<r<k} x_i \neq x_r \right) \Rightarrow \exists y \left[\wedge_{0<i\leq j+1} E(y,x_i) \;\&\; \wedge_{j<i<k} (y \neq x_i \;\&\; \neg E(y,x_i)) \right] \right)$$

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.

$$T_k \equiv \wedge_{0<j<k} S_{k,j}$$

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:

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

Theorem D.1 ([Fag76], [BlHa79]): For any fixed k>0,

$$\lim_{n\to\infty} P_n(T_k) = 1$$

proof: Given j<k, and distinct vertices $x_1 \ldots x_{k-1}$ 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_{kj} is:

$$\left(1 - (1/2^{k-1})\right)^{n-(k-1)} \qquad \leq \quad \alpha^n$$

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

$$n^k \cdot \alpha^k$$

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,

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

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:

$$R = \left\{ (\langle a_1 \ldots a_r \rangle, \langle b_1 \ldots b_r \rangle) \;\middle|\; \begin{array}{l} 0 \leq r \leq k,\ a_i \in G,\ b_i \in H,\ \& \\ \langle a_1 \ldots a_k \rangle \simeq \langle b_1 \ldots b_k \rangle \end{array} \right\}$$

satisfies (a) - (d) of Proposition 4.3. Therefore $G \equiv_{Var[k]} H$. ∎

Corollary D.3: Graph Isomorphism is not in Var(w.o. Suc)[k].

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

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

By Theorem 4.1 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 4.2, $G_k \equiv_{Var[k]} H_k$. 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 right component. Thus, $\langle G_k, H_k \rangle \models F_n$, but G_k is not isomorphic to H_k. This contradiction proves the corollary. ∎

Almost all graphs have a Hamilton circuit; however, in [BlHa79] 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. Suc)[*].

Using similar techniques we can show the following:

Theorem D.5: Clique(k+1) is not in Var(w.o. Suc)[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:

$$\begin{array}{lll} V_n & = & \{ \langle i,j \rangle \mid 1 \leq i \leq k,\ 1 \leq j \leq n \} \\ E_n & = & \{ (\langle i_1, j_1 \rangle, \langle i_2, j_2 \rangle) \mid i_1 \neq i_2 \} \end{array}$$

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 $\lim_{n\to\infty} \text{Prob}(A_n' \models T_k) = 1$. (This follows from the same argument as in the proof of Theorem 4.1, 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. ∎

Section D: Conclusions

We have shown that first order expressibility is a viable view of computational complexity. We feel that it 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 successor) a tractable problem. Furthermore our simulation theorems show that optimal sentences (with successor) for a property C can be easily translated to nearly optimal algorithms for checking C.

The following general areas of exploration are suggested:

(1): Find upper and lower bounds on Var&Sz(w.o. Suc) 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 successor. This seems hard but worthwhile; possible techniques are discussed in [Im80a] and [Im80b].

Acknowledgements

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 the summer of 1980, where and when this paper was completed.

References

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

[BlHa79] : Blass,A., Harary,F., "Properties of Almost All Graphs and Complexes," J. of Graph Theory Vol. 3, 1979, pp. 225-240.

[ChSt76] : Chandra,S., 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.

[Fag76] : _____, "Probabilities on Finite Models," JSL Vol 41, No. 1, 1976, pp. 50-58.

[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.

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

[Im80a] : _____, "First Order Expressibility as a New Complexity Measure," Ph.D. Thesis, Cornell University, August, 1980.

[Im80b] : _____, "Number of Quantifiers is Better than Number of Tape Cells," to appear in JCSS, 1980.

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

[Rei79] : Reif,J., "Universal Games of Incomplete Information," Proc. 11th SIGACT, 1979, pp. 288-308.

[Ru79a] : Ruzzo,W, "Tree-Size Bounded Alternation," Proc. 11th SIGACT, 1979, pp. 352-359.

[Ru79b] : _____, "On Uniform Circuit Complexity," Proc. 20th FOCS, 1979, pp. 312-318.

[Sav70] : Savitch,W., "Maze Recognizing Automata and Nondeterministic Tape Complexity," JCSS 7, 1973, pp. 389-403.

[Sud78]: Sudborough,I., 'On the Tape Complexity of Deterministic CFL's," JACM Vol. 25, No. 3, 1978, pp. 405-414.