Number of Variables Is Equivalent To Space

Neil Immerman*

University of Massachusetts, Amherst

immerman@cs.umass.edu

Jonathan F. Buss†

University of Waterloo

jfbuss@math.uwaterloo.ca

David A. Mix Barrington

University of Massachusetts, Amherst

barring@cs.umass.edu

March 21, 2000

Abstract

We prove that the set of properties describable by a uniform sequence of first-order sentences using at most $k+1$ distinct variables is exactly equal to the set of properties checkable by a Turing machine in $DSPACE[n^k]$ (where n is the size of the universe). This set is also equal to the set of properties describable using an iterative definition for a finite set of relations of arity k. This is a refinement of the theorem $PSPACE = VAR[O[1]]$ [7]. We suggest some directions for exploiting this result to derive trade-offs between the number of variables and the quantifier depth in descriptive complexity.

1 Introduction

In Descriptive Complexity one analyzes the complexity of a language in terms of the complexity of describing the language. It is known that the quantifier depth and number of variables needed to express the membership property of a language is closely related to the

*Research supported in part by NSF grant CCR-9505446.

†Research supported in part by NSERC, and performed while the author was visiting the University of Massachusetts, Amherst.

parallel time and amount of hardware needed to check whether an input is in the language. For a long time, the basic question of complexity—namely what are the trade-offs between time and hardware—has remained open. We have been attempting to understand this question in terms of the trade-off between number of variables and quantifier depth. In this paper we tighten the known relationship between number of variables and deterministic space.

This paper is organized as follows. Section 2 gives the relevant background, definitions, and references. Section 3 proves our main result. Section 4 includes potential applications of this result and suggests some attacks on the difficult questions in complexity theory, from the descriptive point of view.

2 Background and Definitions

In this section we sketch the relevant background and definitions. More detail can be found in [12].

2.1 First-Order Logic

A *vocabulary* $\tau = \langle R_1^{a_1}, \ldots, R_k^{a_k}, c_1, \ldots, c_r \rangle$ is a tuple of relation symbols and constant symbols, where each $R_i^{a_i}$ is a relation symbol of arity a_i. In the sequel we will often omit the superscripts to improve readability. A finite *structure* with vocabulary τ is a tuple $\mathcal{A} = \langle \{0, 1, \ldots, n-1\}, R_1^{\mathcal{A}}, \ldots, R_k^{\mathcal{A}}, c_1^{\mathcal{A}}, \ldots, c_r^{\mathcal{A}} \rangle$, consisting of a universe $U^{\mathcal{A}} = \{0, \ldots, n-1\}$ and relations $R_1^{\mathcal{A}}, \ldots, R_k^{\mathcal{A}}$ of arities $a_1, \ldots, a_k$ on $U^{\mathcal{A}}$ corresponding to the relation symbols $R_1^{a_1}, \ldots, R_k^{a_k}$ of τ, and constants $c_1^{\mathcal{A}}, \ldots, c_r^{\mathcal{A}}$ from $U^{\mathcal{A}}$ corresponding to the constant symbols $c_1, \ldots, c_r$ from τ. We write $|\mathcal{A}|$ to denote $U^{\mathcal{A}}$, the universe of $\mathcal{A}$, and we write $\|\mathcal{A}\|$ to denote n, the cardinality of the universe. Let $STRUC[\tau]$ denote the set of all (finite) structures of vocabulary τ.

For example, if τ_g consists of a single binary relation symbol E (standing for edge) then a structure $G = \langle \{0, \ldots, n-1\}, E \rangle$ with vocabulary τ_g is a graph on n vertices. Similarly if τ_s consists of a single unary relation symbol S, then a structure $S = \langle \{0, \ldots, n-1\}, S \rangle$ with vocabulary τ_s is a binary string of length n.

Let the symbol '$\leq$' denote the usual ordering on the natural numbers. We will include $\leq$ as a logical relation in our first-order languages. This seems necessary in order to simulate machines whose inputs are structures given in some order. For convenience we also include the constant symbols 0 and *max* referring to the first and last elements of the structure respectively, and the logical relation $s(x, y)$ true whenever y is the immediate successor of x in the ordering $\leq$. For technical reasons, we also include the logical relation BIT, where $BIT(x, y)$ holds iff bit x of the binary expansion of y is 1.

We define the *first-order language* $\mathcal{L}(\tau)$ to be the set of formulas built up from the relation and constant symbols of τ, and the logical relation symbols and constant symbols: $=, \leq, s, \mathrm{BIT}, 0, max$, using logical connectives: $\wedge, \vee, \neg$, variables: $x, y, z, \ldots$, and quantifiers: $\forall, \exists$. A *sentence* is a formula with no free variables. Every sentence $\varphi \in \mathcal{L}(\tau)$ is either true or false in any structure $\mathcal{A} \in STRUC[\tau]$. We write $\mathcal{A} \models \varphi$ to mean that $\mathcal{A}$ satisfies φ. Let

MOD$[\varphi]$ denote the set of all models of φ:

$$\mathrm{MOD}[\varphi] \;=\; \{\, \mathcal{A} \in \mathit{STRUC}[\tau] \mid \mathcal{A} \models \varphi \,\}.$$

Here either τ is previously specified, or it is the vocabulary of all nonlogical symbols occurring in φ.

We will think of a *problem* as a set of structures of some vocabulary τ. It suffices to only consider problems on binary strings, but it is more interesting to be able to talk about other vocabularies, e.g. graph problems, as well. For definiteness, we will fix a scheme for coding an input structure as a binary string. If $\mathcal{A} = \langle\{0,1,\ldots,n-1\}, R_1^{\mathcal{A}},\ldots,R_k^{\mathcal{A}}, c_1^{\mathcal{A}},\ldots,c_r^{\mathcal{A}}\rangle$, is a structure of vocabulary τ, then $\mathcal{A}$ will be encoded as a binary string $\mathrm{bin}(\mathcal{A})$ of length $I_\tau(n) = n^{a_1} + \cdots + n^{a_k} + r\lceil\log n\rceil$, consisting of one bit for each a_i-tuple, potentially in the relation R_i, and $\lceil \log n\rceil$ bits to name each constant, c_j. (In this special case that the vocabulary is empty we encode a structure of size n as the string of n zeros.)

Define the complexity class FO to be the set of all first-order expressible problems. FO is a uniform version of the circuit class AC^0 [1] and it is equal to the set of problems acceptable in constant time on a polynomial size concurrent parallel random access machine [10].

On notation: We reserve n to indicate the size of the universe of the input structure. We will denote the length of the input string by $\hat{n} = I_\tau = |\mathrm{bin}(\mathcal{A})|$. The length $\hat{n}$ is polynomially related to n, and in the case where τ consists of a single unary relation (and thus inputs are binary strings) we have $\hat{n} = I_{\tau_s}(n) = n$. In the case of graphs, $\hat{n} = I_{\tau_g}(n) = n^2$. Our main result says that $\mathit{DSPACE}[n^k] = \mathit{VAR}[k+1]$. Thus, for binary-string problems this would imply that $\mathit{DSPACE}[\hat{n}] = \mathit{DSPACE}[n] = \mathit{VAR}[2]$. However for graph problems, $\mathit{DSPACE}[\hat{n}] = \mathit{DSPACE}[n^2] = \mathit{VAR}[3]$. (In our notation a graph algorithm that uses space linear in the size of the adjacency matrix is a $\mathit{DSPACE}[n^2]$ algorithm.) Since variables are fixed to range over the size-n universe, a standard first-order variable always describes $\log n$ bits of information.

2.2 Inductive Definitions

A useful way to increase the power of first-order logic without jumping all the way up to second order logic is to add the power to define new relations by induction. For example, consider the vocabulary $\tau_g = \langle E\rangle$ of graphs. We can define the reflexive, transitive closure E^* of E as follows. Let R be a binary relation variable, and consider the formula

$$\varphi(R,x,y) \equiv (x=y) \;\vee\; \exists z(E(x,z) \wedge R(z,y)) \tag{2.1}$$

The formula φ gives an inductive definition of E^* which may be more suggestively written as

$$E^*(x,y) \equiv (x=y) \;\vee\; \exists z(E(x,z) \wedge E^*(z,y)). \tag{2.2}$$

For any structure $\mathcal{A}$ with vocabulary τ_g, φ induces a map from binary relations on the universe of $\mathcal{A}$ to binary relations on the universe of $\mathcal{A}$,

$$\varphi_{\mathcal{A}}(R) \;=\; \{\, \langle a,b\rangle \mid \mathcal{A} \models \varphi(R,a,b) \,\}$$

Such a map is called *monotonic* if for all R, S,

$$R \subseteq S \;\Rightarrow\; \varphi_{\mathcal{A}}(R) \subseteq \varphi_{\mathcal{A}}(S).$$

Note that since R appears only positively in φ in Equation 2.1, $\varphi_{\mathcal{A}}$ is monotonic. Let $\varphi_{\mathcal{A}}^r$ denote $\varphi_{\mathcal{A}}$ iterated r times. With φ defined as in Equation 2.1, $\mathcal{A}$ any graph, and $r \geq 0$, observe that

$$\varphi_{\mathcal{A}}^r(\emptyset) \;=\; \{\, \langle a, b\rangle \in |\mathcal{A}|^2 \mid \text{distance}(a, b) \leq r - 1\,\}.$$

In particular, if $n = \|\mathcal{A}\|$, then $\varphi_{\mathcal{A}}^n(\emptyset) = E^*$ is the least fixed point of $\varphi_{\mathcal{A}}$, *i.e.*, the minimal relation T such that $\varphi_{\mathcal{A}}(T) = T$. This is a general situation as we now show in the finite version of the Knaster-Tarski Theorem.

Theorem 2.3 (Knaster and Tarski [15, 14]) *Let R be a new relation symbol of arity k, and let $\psi(R, x_1, \ldots, x_k)$ be an R-positive first-order formula (*i.e.*, R occurs only within an even number of negation signs). Then for any finite structure $\mathcal{A}$, the least fixed point of $\psi_{\mathcal{A}}$ exists and is equal to $\psi_{\mathcal{A}}^r(\emptyset)$, where r is minimal so that $\psi_{\mathcal{A}}^r(\emptyset) = \psi_{\mathcal{A}}^{r+1}(\emptyset)$.*

Proof Since ψ is R-positive, $\psi_{\mathcal{A}}$ is monotonic. Each application of $\psi_{\mathcal{A}}$ at a non-fixed point adds at least one new k-tuple to the relation. Since there are only n^k possible k-tuples, a fixed point must be reached with $r \leq n^k$ applications.

Thus $\psi_{\mathcal{A}}^r(\emptyset)$ is a fixed point of $\psi_{\mathcal{A}}$. Let T be any other fixed point, and let $R_i = \psi_{\mathcal{A}}^i(\emptyset)$ for $0 \leq i \leq r$. It is easy to see by induction on i that $R_i \subseteq T$. Of course $R_0 = \emptyset \subseteq T$. Assuming $R_i \subseteq T$, the monotonicity of $\psi_{\mathcal{A}}$ implies that $\psi_{\mathcal{A}}(R_i) \subseteq \psi_{\mathcal{A}}(T)$, i.e. $R_{i+1} \subseteq T$. Thus $\psi_{\mathcal{A}}^r(\emptyset)$ is the least fixed point as claimed. $\square$

Theorem 2.3 tells us that any R^k-positive formula $\varphi(R^k, x_1, \ldots, x_k)$ determines a k-ary least-fixed-point relation. We will write $(\mathit{LFP}_{R^k x_1 \ldots x_k}\varphi)$ to denote this least fixed point. The least-fixed-point operator (LFP) thus formalizes the definition of new relations by induction.

Definition 2.4 Define $\mathit{FO}(\mathit{LFP})$, the language of first-order inductive definitions, by adding a least-fixed-point operator to first-order logic. If $\varphi(R^k, x_1, \ldots, x_k)$ is an R^k-positive formula in $\mathit{FO}(\mathit{LFP})$, then $(\mathit{LFP}_{R^k x_1 \ldots x_k}\varphi)$ is a formula in $\mathit{FO}(\mathit{LFP})$ denoting the least fixed point of φ. $\square$

Immerman and Vardi independently characterized the complexity of $\mathit{FO}(\mathit{LFP})$ as follows.

Theorem 2.5 ([8, 16]) $\mathit{FO}(\mathit{LFP}) = P$

The number of iterations until an inductive definition closes is called its *depth.*[1] Inductive depth turns out to be linearly related to parallel time, cf. Theorem 2.11.

[1]In the logical literature where structures are usually infinite this is called the *closure ordinal*, cf. [14].

Definition 2.6 Let $\varphi(R^k, x_1, \ldots, x_k)$ be an R-positive formula, where R is a relation symbol of arity k, and let $\mathcal{A}$ be a structure. Define the *depth* of φ in $\mathcal{A}$, denoted $|\varphi_{\mathcal{A}}|$, to be the minimum r such that

$$\mathcal{A} \models \Big(\varphi^r(\emptyset) \leftrightarrow \varphi^{r+1}(\emptyset)\Big).$$

As we saw in the proof of Theorem 2.3, $|\varphi_{\mathcal{A}}| \le n^k$. Define the depth of φ as a function of n equal to the maximum depth of φ in $\mathcal{A}$ for any structure $\mathcal{A}$ of size n:

$$|\varphi|\,(n) \;=\; \max\Big\{\, |\varphi_{\mathcal{A}}| \;\Big|\; \|\mathcal{A}\| = n \,\Big\}.$$

□

Definition 2.7 Let $IND[f(n)]$ be the sublanguage of $FO(LFP)$ in which we only include least fixed points of first-order formulas φ for which $|\varphi|$ is $O[f(n)]$.[2] □

2.3 Iterating First-Order Formulas

Theorem 2.3 shows that the least-fixed-point operator amounts to a polynomial iteration operator. This is even more apparent when we put the inductive definitions into the following simple normal form. (The notation $(\forall x\,.\,M)\psi$ means $(\forall x)(M \to \psi)$, and $(\exists x\,.\,M)\psi$ means $(\exists x)(M \wedge \psi)$.)

Lemma 2.8 ([14, 8]) *Let φ be an R-positive first-order formula. Then φ can be written in the form*

$$\varphi(R, x_1, \ldots, x_k) \equiv (Q_1 z_1\,.\,M_1)\ldots(Q_s z_s\,.\,M_s)(\exists x_1 \ldots x_k\,.\,M_{s+1})R(x_1, \ldots, x_k)$$

where the M_i's are quantifier-free formulas in which R does not occur.

Proof This is a straightforward induction on the structure of φ. We will show that the lemma holds with M_{s+1} in the following restricted form.

$$M_{s+1} \;\equiv\; (x_1 = z_{i_1} \;\wedge\; x_2 = z_{i_2} \;\wedge\; \ldots \;\wedge\; x_k = z_{i_k}).$$

The only nontrivial case is the inductive step when $\varphi = \alpha \wedge \beta$; suppose

$$\begin{aligned} \alpha &\;\equiv\; (Q_1 y_1\,.\,N_1)\ldots(Q_t y_t\,.\,N_t)(\exists x_1 \ldots x_k\,.\,N_{t+1})R(x_1, \ldots, x_k) \\ \beta &\;\equiv\; (Q_1 z_1\,.\,M_1)\ldots(Q_s z_s\,.\,M_s)(\exists x_1 \ldots x_k\,.\,M_{s+1})R(x_1, \ldots, x_k) \end{aligned}$$

where we may assume that the y's and z's are disjoint. Let

$$\begin{aligned} QB_1 &\;\equiv\; (Q_1 y_1\,.\,N_1)\ldots(Q_t y_t\,.\,N_t)\;, \\ QB_2 &\;\equiv\; (Q_1 z_1\,.\,M_1)\ldots(Q_s z_s\,.\,M_s). \end{aligned}$$

[2] Of course it may not be decidable whether a given formula is in some $IND[f(n)]$. This is analogous to the definition of complexity classes where it is not decidable if a given Turing machine always runs in a given time or space bound.

Let $\psi(\bar{u}/\bar{x})$ denote the formula ψ with variables $u_1, \ldots u_k$ substituted for $x_1 \ldots x_k$ and define the quantifier-free formulas,

$$\begin{array}{rcl} S & \equiv & (b = 0 \ \wedge \ N_{t+1}(\bar{u}/\bar{x})) \ \vee \ (b = 1 \ \wedge \ M_{s+1}(\bar{u}/\bar{x})) \ , \\ T & \equiv & (u_1 = x_1 \ \wedge \ \ldots \ \wedge \ u_k = x_k). \end{array}$$

In this case we have that

$$\varphi \ \equiv \ (\forall b.b \le 1)(QB_1)(QB_2)(\exists \bar{u} \, . \, S)(\exists \bar{x} \, . \, T)R(x_1, \ldots, x_k).$$

□

The above requantification of the x_i's means that these variables may occur freely in $M_1 \ldots M_s$, but they are bound in M_{s+1} and $R(x_1, \ldots, x_k)$. The same variables may now be requantified. We write QB to denote the quantifier block $(Q_1 z_1 \, . \, M_1) \ldots (Q_s z_s \, . \, M_s)(\exists x_1 \ldots x_k \, . \, M_{s+1})$. Thus, in particular, for any structure $\mathcal{A}$, and any $r \in \mathbb{N}$,

$$\mathcal{A} \models (\varphi^r_{\mathcal{A}}(\emptyset)) \leftrightarrow ([QB]^r(\mathit{false})) \ .$$

Here $[QB]^r$ means QB repeated r times (literally). It follows immediately that if $t = |\varphi|(n)$, and $\mathcal{A}$ is any structure of size n then

$$\mathcal{A} \models (\mathit{LFP} \ \varphi) \leftrightarrow ([QB]^t(\mathit{false})).$$

We define $FO[t(n)]$ to be the set of properties defined by quantifier blocks iterated $t(n)$ times.

Definition 2.9 A set $C \subseteq STRUC[\tau]$ is a member of $FO[t(n)]$ iff there exist quantifier-free formulas M_i (all $0 \le i \le k$) from $\mathcal{L}(\tau)$, a tuple $\bar{c}$ of constants, and a quantifier block

$$QB \ = \ [(Q_1 x_1 \, . \, M_1) \ldots (Q_k x_k \, . \, M_k)]$$

such that if we let $\varphi_n = [QB]^{t(n)} M_0(\bar{c})$ (all $n \ge 1$), then for all $\mathcal{A} \in STRUC[\tau]$ with $\|\mathcal{A}\| = n$,

$$\mathcal{A} \in C \ \Leftrightarrow \ \mathcal{A} \models \varphi_n.$$

□

In the proof of Lemma 2.8 we introduced quantified boolean variables into the quantifier block to replace logical "and"s and "or"s. In Theorem 3.1 we will be carefully counting the number of domain variables. The following lemma shows that the number of domain variables need not be increased to take care of conjunctions and disjunctions.

Lemma 2.10 *Suppose that we have two quantifier blocks with identical domain quantifiers in identical order (ignoring any boolean quantifiers):*

$$\begin{array}{rcl} QB_1 & = & [(Q_1 v_1 \, . \, M_1) \ldots (Q_s v_s \, . \, M_s)] \\ QB_2 & = & [(Q_1 v_1 \, . \, N_1) \ldots (Q_s v_s \, . \, N_s)] \end{array}$$

Then the conjunction and disjunction of these quantifier blocks may be written in the same form.

Proof For example, the conjunction can be written with an additional universally quantified boolean variable as

$$[QB_1]\varphi \wedge [QB_2]\varphi \;\equiv\; [(\forall b)(Q_1 v_1 . R_1) \ldots (Q_s v_s \, . \, R_x)]\varphi$$

where

$$R_i \;=\; (b = 0 \;\wedge\; M_i) \;\vee\; (b = 1 \;\wedge\; N_i).$$

□

Inductive depth and first-order iterations are intimately connected with parallel complexity. Let $CRAM[t(n)]$ denote the set of problems accepted by concurrent, parallel random access machines in parallel time $O[t(n)]$ using polynomially much hardware.[10]

Theorem 2.11 ([10]) *For all polynomially bounded and constructible $t(n)$,*

$$CRAM[t(n)] \;=\; IND[t(n)] \;=\; FO[t(n)].$$

Example 2.12 ([10])We show how to transfer a $\log n$ depth inductive definition of the transitive closure of a graph to an equivalent $FO[\log n]$ definition.

Let E be the edge predicate for a graph G with n vertices. We can inductively define E^*, the reflexive, transitive closure of G, by

$$E^*(x,y) \;\equiv\; x = y \;\vee\; E(x,y) \;\vee\; (\exists z)(E^*(x,z) \wedge E^*(z,y)).$$

Let $P_n(x,y)$ mean that there is a path of length at most n from x to y. Then we can rewrite the above definition of E^* as

$$\begin{aligned} P_n(x,y) \;&\equiv\; x = y \;\vee\; E(x,y) \;\vee\; (\exists z)(P_{n/2}(x,z) \wedge P_{n/2}(z,y)) \\ &\equiv\; (\forall z \, . \, M_1)(\exists z)(P_{n/2}(x,z) \wedge P_{n/2}(z,y)), \end{aligned}$$

where $M_1 \equiv \neg(x = y \;\vee\; E(x,y))$. Note that there is no free occurrence of the variable z after the $\forall z$ quantifier. Thus in this case $(\forall z \, . \, M_1)\alpha$ is equivalent to $(M_1 \rightarrow \alpha)$. Next,

$$P_n(x,y) \;\equiv\; (\forall z \, . \, M_1)(\exists z)(\forall uv \, . \, M_2)(P_{n/2}(u,v)),$$

where $M_2 \equiv (u = x \;\wedge\; v = z) \;\vee\; (u = z \;\wedge\; v = y)$. Now,

$$P_n(x,y) \;\equiv\; (\forall z \, . \, M_1)(\exists z)(\forall uv \, . \, M_2)(\forall xy \, . \, M_3)(P_{n/2}(x,y)) \;,$$

where $M_3 \equiv (x = u \;\wedge\; y = v)$. Thus,

$$P_n(x,y) \;\equiv\; [QB]^{\lceil \log n \rceil}(P_1(x,y)),$$

where $QB = (\forall z \, . \, M_1)(\exists z)(\forall uv \, . \, M_2)(\forall xy \, . \, M_3)$. Note that

$$P_1(x,y) \;\equiv\; [QB](\mathit{false}).$$

It follows that

$$P_n(x,y) \;\equiv\; [QB]^{\lceil 1+\log n \rceil}(\mathit{false}) \;,$$

and thus $E^* \in FO[\log n]$ as claimed. □

2.4 Unbounded Iterations

As we saw in Theorem 2.3, inductive definitions must close after at most polynomially many steps because of their monotonicity. Now we generalize inductive definitions to *iterative definitions* in which the requirement of monotonicity is removed.[3]

Definition 2.13 Define $ITER[t(n), \mathit{arity}\ k]$ to be the set of properties definable by iterating $t(n)$ times, the simultaneous first-order definitions of a set of c relations of arity k, for some constant c. Of course, after $t(n) = 2^{cn^k}$ iterations, these relations will either reach a fixed point, or be in a cycle. Thus, define

$$ITER[\mathit{arity}\ k] \;=\; \bigcup_{c=1}^{\infty} ITER[2^{cu^k}, \mathit{arity}\ k]$$

$\square$

Similarly, we define arbitrary iterations of quantifier blocks.

Definition 2.14 Define $FO[t(n)]\text{-}VAR[k]$ to be the set of properties definable in the form

$$[QB]^{t(n)} M_0$$

for some quantifier block QB containing domain variables from the set $\{x_1, \ldots, x_k\}$, and some boolean variables $b_1, \ldots, b_l$. The formula M_0 contains only boolean variables. As in the definition of $ITER[\mathit{arity}\ k]$, the truth assignment of all the variables will cycle or stabilize after at most $t(n) = 2^{cn^k}$ iterations. Thus, define

$$VAR[k+1] = \bigcup_{c=1}^{\infty} FO[2^{cn^k}]\text{-}VAR[k+1]$$

$\square$

We can extend the above definition of $FO[\cdot]\text{-}VAR[\cdot]$ by allowing restricted domain variables. Let $f : \mathbb{Z}^+ \to \mathbb{N}$ be such that for all $n \in \mathbb{Z}^+$, $f(n-1) < n^k$. In this case, for $c \in \{0, 1, \ldots, n-1\}$, $f(c)$ can be represented as a k-tuple of elements from $\{0, 1, \ldots, n-1\}$ in n-ary notation.

We say that f is *first-order representable* iff there exists a first-order formula B_f using only the numeric relations and constants: $\leq, BIT, 0, \mathit{max}$ such that for all structures $\mathcal{A}$ and k-tuples $\bar{a} \in |\mathcal{A}|^k$,

$$\mathcal{A} \models \mathcal{B}_f(\bar{a}) \qquad \Leftrightarrow \qquad \bar{a} \leq f(\|\mathcal{A}\| - 1),$$

that is $\varphi_f(\bar{x})$ says that $\bar{x} \leq f(\mathit{max})$.

As examples, the functions $f(n) = n+1, 2n, n^2, \lceil \log n \rceil$ are all first-order representable [12]. The following formula represents $\lceil \log(n) \rceil$:

$$B_{\lceil \log \rceil}(x) \quad \equiv \quad (\exists y)(BIT(\mathit{max}, y) \wedge (x \leq y \vee s(y, x)))$$

[3]This is equivalent to the addition of a "while" operator [16].

A *restricted* variable is one that only occurs bound by a quantifier of the form $(Qx\,.\,(B_f(x))\wedge M)$, where B_f is a bounding formula. The size of a restricted variable is $\lfloor\log(f(max))\rfloor+1$, the number of bits required to write its maximum value.

Let $FO[t(n)]\text{-}VAR[k;r]$, for $r\le\log n$, be the set of properties definable as above with a quantifier block QB containing at most k unrestricted domain variables and some restricted domain variables whose total size is at most $r+O(1)$ bits. Define

$$VAR[k+1;r]=\bigcup_{c=1}^{\infty}FO[2^{c2^r n^k}]\text{-}VAR[k+1;r].$$

In a similar way, we define $ITER[\text{arity } k;r]$ to be the set of queries definable by an iterative definition of a relation of arity $k+1$ whose last argument is bounded to values of size at most $O(2^r)$, i.e., those that can be described using $r+O(1)$ bits.

3 The Equivalence of Space to Number of Variables

In this section we prove our main theorem, relating the descriptive complexity of a problem to its computational complexity. As usual, we encode structures as strings when considering them as input to a Turing machine. A structure of cardinality n and maximum arity a has an encoding of length $\hat{n}=I_\tau(n)=\Theta(n^a)$.[4]

Theorem 3.1 *For any space bound $s(n)$ satisfying $\log n\le s(n)\le n^{O(1)}$, let $k=\lfloor\log_n(s(n))\rfloor$. If $\hat{n}\le O(n^{k+1})$ then,*

$$DSPACE[s(n)] \;=\; VAR[k+1;\log(s(n)/n^k)] \;=\; ITER[arity\ k;\log(s(n)/n^k)]$$

The condition that $\hat{n}\le O(n^{k+1})$ — equivalently $a\le k+1$ — is needed so that the $VAR[k{+}1;\log(s(n)/n^k)]$ formula may read its input. Note that when we restrict our attention to string problems, $\tau=\tau_s$, $a=1$ and thus this condition always holds.

Restricting to the case that $s(n)$ is a power of n we have,

Corollary 3.2 *For any $k=1,2,\ldots$, if $\hat{n}\le O(n^{k+1})$, i.e., the arity of the input relations is at most $k+1$, then*

$$DSPACE[n^k]=VAR[k+1]=ITER[arity\ k]\ .$$

Another interesting special case occurs when $s(n)<n$ and thus $k=0$. In this case, $DSPACE[[]s(n)]$ consists of queries expressible with one ordinary domain variable and one restricted to $\log(s(n))$ bits. This is also equal to those iteratively defined relations of arity one over a variable whose domain is limited to size $s(n)$.

We first prove Corollary 3.2. We prove three lemmas showing the following containments:

$$DSPACE[n^k]\subseteq VAR[k+1]\subseteq ITER[arity\ k]\subseteq DSPACE[n^k]\ .$$

Then we will show how to generalize to obtain a proof of Theorem 3.1.

[4]Thus to interpret Theorem 3.1 using input length instead of n, we can replace $s(n)$ by $s(\hat{n})$ and n by $\hat{n}^{1/a}$.

Lemma 3.3 $DSPACE[n^k] \subseteq VAR[k+1]$.

Proof Let M be a $DSPACE[n^k]$ Turing machine. M's work space consists of n^k tape cells each of which holds a symbol from some finite alphabet, Σ.

The contents of M's tape at time $t+1$ are a deterministic, local transformation of the contents at time t. Namely, the contents of cell p at time $t+1$ is a function of the contents of cells $p-1, p, p+1$ at time t.

We write a logical formula $C_t(\bar{x}, \bar{b})$ meaning that after step t of M's computation, the cell at position $\bar{x}$ is $\bar{b}$. Here $\bar{x} = x_1, \ldots, x_k$ is a k-tuple of variables ranging over the set $\{0, \ldots, n-1\}$ and $\bar{b}$ is a tuple of boolean variables coding an element of Σ.

The following is an iterative definition of C_t:

$$C_{t+1}(\bar{x}, \bar{b}) \quad \equiv \quad \bigvee_{\langle \bar{a}_{-1}, \bar{a}_0, \bar{a}_1 \rangle \to \bar{b}} \Big(C_t(\bar{x}-1, \bar{a}_{-1}) \wedge C_t(\bar{x}, \bar{a}_0) \wedge C_t(\bar{x}+1, \bar{a}_1) \Big) \tag{3.4}$$

Here, the disjunction is over the finite set of quadruples $(\bar{a}_{-1}, \bar{a}_0, \bar{a}_1, \bar{b})$ such that the first three symbols lead to the fourth symbol in one move of M. Note that this set of quadruples is exactly a representation of M's state table.

It is straightforward to write C_0 with k domain variables. Our assumption that $\hat{n} \le O(n^k)$ means that the input fits on the work tape. To tell whether $C_0(x_1, x_2, \bar{b})$ holds, we just have to know what the input symbol would be at location $nx_1 + x_2$. For example, if the input is a graph, then $C_0(x_1, x_2, \bar{b})$ would hold if $E(x_1, x_2)$ holds and the booleans $\bar{b}$ encode the tape symbol "1", or if $E(x_1, x_2)$ does not hold and $\bar{b}$ encodes the type symbol "0".

Furthermore, M accepts its input iff it eventually reaches its accept state. Let $\bar{1}$ code the appropriate accept symbol. Thus M accepts its input iff eventually $C_t(\bar{0}, \bar{1})$ holds.

The lemma will be proved once we show the following:

Claim 3.5 *There is a quantifier block QB containing $k+1$ domain variables such that Equation (3.4) may be rewritten as:* $C_{t+1}(\bar{x}, \bar{b}) \equiv [QB] C_t(\bar{x}, \bar{b})$.

The proof of Claim 3.5 is purely symbol manipulation. We first write quantifier blocks QB_+ and QB_- whose job it is to replace $\bar{x}$ by $\bar{x}+1$ and $\bar{x}-1$ respectively, i.e., for any formula φ, we have,

$$\begin{aligned} \varphi(\bar{x}+1) &\equiv [QB_+]\varphi(\bar{x}) \ , \\ \varphi(\bar{x}-1) &\equiv [QB_-]\varphi(\bar{x}) \ . \end{aligned}$$

These quantifier blocks can be written with $k+1$ domain variables. The idea is to add one to $\bar{x}$ by replacing x_k with its successor, or, if $x_k = max$, by replacing x_k by 0 and x_{k-1} by its successor, or, etc. We existentially quantify a tuple of boolean variables, $\bar{c}$, to guess for which i, $1 \le i \le k$, x_i will be incremented. For $j > i$, it must be that $x_j = max$ and $x'_j = 0$.

The form of the quantifier block will be as follows,

$$(\exists \bar{c}.P)(\exists y.N_k)(\exists x_k.M_k)(\exists y.N_{k-1})(\exists x_{k-1}.M_{k-1}) \cdots (\exists y.N_1)(\exists x_1.M_1) \ .$$

The quantifier-free conditions P, N_i and M_i are as follows. Here "s" is the successor relation.

$$\begin{aligned} P &\equiv \bigvee_{i=1}^{k} (\bar{c} = i \ \wedge \ x_i \neq \max \wedge x_{i+1} = x_{i+2} = \cdots = x_k) \\ N_i &\equiv (i < \bar{c} \ \wedge \ y = x_k) \quad \vee \quad (i = \bar{c} \ \wedge \ s(x_k, y)) \\ &\qquad \vee \quad (i > \bar{c} \ \wedge \ y = \max) \\ M_i &\equiv x_i \ = \ y \ . \end{aligned}$$

Thus, we have QB_+, QB_-, and, trivially, QB_0, Observe that the desired QB of Claim 3.5 is a positive boolean combination of these three quantifier blocks. It follows from Lemma 2.10 that QB exists and has $k+1$ domain variables, as desired. This completes the proof of the Claim and thus of Lemma 3.3. □

Lemma 3.6 *$VAR[k+1] \ \subseteq \ ITER[arity \ k]$.*

Proof Here we have a quantifier block of the form

$$QB \ = \ [(Q_1 x_{i_1}.M_1)(B_1)\dots(Q_r x_{i_r}.M_r)(B_r)],$$

where each $i_j \in \{1, \dots, k+1\}$, the M_i are quantifier-free, and the B_i are blocks of boolean quantifiers over boolean variables $\{b_1, \dots, b_c\}$. We can convert the iteration of QB into an iterative definition of a relation R of arity k. R takes as arguments k domain elements, plus a bounded number of boolean arguments.

The reason that arity k suffices is that the variable x_{i_1} does not occur freely in a formula beginning $(Q_1 x_{i_1}.M_1)$. Thus, our iterative definition is essentially,

$$R(x_2, \dots, x_k; b_1, \dots, b_c) \quad \equiv \quad [QB]R(x_2, \dots, x_k; b_1, \dots, b_c) \tag{3.7}$$

The only problem with Equation 3.7 is that we haven't said how to begin. The answer is that at the beginning we use $[QB]M_0$ and every other time we use Equation 3.7. Thus, the iterative definition is

$$\begin{aligned} R(x_2, \dots, x_k; \bar{b}) \ \equiv \ & (\forall \bar{x})(\forall \bar{b})(\neg R(\bar{x}, \bar{b})) \rightarrow [QB]M_0 \ \wedge \\ & (\exists \bar{x})(\exists \bar{b})(R(\bar{x}, \bar{b})) \rightarrow [QB]R(\bar{x}, \bar{b}) \end{aligned} \tag{3.8}$$

It is immediate that the iterative definition of R captures the meaning of the iterated quantifier block and that it has arity k. Notice that allowing extra boolean variable arguments in R means that we only need a single iterative definition. Without this feature we would need to define 2^c relations simultaneously, corresponding to all the possible values of the booleans $b_1, \dots b_c$. □

Lemma 3.9 *$ITER[arity \ k] \ \subseteq \ DSPACE[n^k]$.*

This last inclusion is obvious because $O[n^k]$ bits suffice to record the current meaning of the bounded number of relations of arity k. Each bit of each relation in the next iteration may then be computed by evaluating a fixed first-order formula. This can be done in $DSPACE[\log n]$ and thus certainly in $DSPACE[n^k]$. This completes the proof of Corollary 3.2.

To complete the proof of Theorem 3.1, we have to do two things:

1. Extend the proof to the case where $s(n) < \hat{n} \leq O(n^{k+1})$. That is the input is longer than the worktape. In this case we need a separate read-only input tape.

2. Extend the proof to the case where $s(n)$ is not an exact power of n.

It is easy to see that Lemmas 3.6 and 3.9 are unaffected by (1), and easily modified to allow (2). It thus suffices to reprove the following strengthening of Lemma 3.3.

Lemma 3.10 *For any space bound $s(n)$ satisfying $\log n \leq s(n) \leq n^{O(1)}$, let $k = \lfloor \log_n(s(n)) \rfloor$. If $\hat{n} \leq O(n^{k+1})$ then,*

$$DSPACE[s(n)] \quad \subseteq \quad VAR[k+1; \log(s(n)/n^k)]$$

Proof Let M be a Turing machine using space $s(n)$. We assume that M has a read-only input tape and that the location of the read head, h, is included on a special section of the work tape. Note that $\log(\hat{n}) = O(\log n) \leq O(s(n))$, so keeping the head position does not affect the space bound.

As in the proof of Lemma 3.3, we will write a logical formula $C_t(\bar{x}, \bar{b})$ meaning that after step t of M's computation, the cell at position $\bar{x}$ has symbol $\bar{b}$. Here $\bar{x}$ is a k-tuple of domain variables plus one variable restricted to size s/n^k, and $\bar{b}$ is a tuple of Boolean variables coding an element of Σ. We will also write $H_t(i)$ meaning that bit i of the read head position at time t is a one. Here i is a variable restricted to be less than $\log(\hat{n})$, thus using $\log\log n$ bits.

It is straightforward to write C_0 and H_0 representing the input configuration, using just the variables $\bar{x}$ and $\bar{b}$. We simply encode the facts that the work tape is blank, M is in its start state and $h = 0$.

We now want to write the iterative definition of C_{t+1} and H_{t+1} using C_t and H_t. The new work is to write the formula ONE_t meaning that the bit being scanned by the read head at time t is a one.

Claim 3.11 *If $\hat{n} \leq O(ns(n)/\log(n)$ then ONE_t is expressible in $VAR[k+1; \log(s/n^k)]$ using C_t and H_t.*

To prove Claim 3.11 let us first assume that $\tau = \tau_s$, i.e., there is a single unary input relation symbol, S. In this case,

$$\mathrm{ONE}_t \quad \equiv \quad (\exists y) \text{ ``}y\text{ is the position of the read head at time }t\text{''} \quad \wedge \; S(y)$$

Where,

$$\text{``}y\text{ is the position of the read head at time }t\text{''} \;\equiv\; (\forall i.B_{\log(n)}(i))(BIT(y,i) \leftrightarrow H_t(i))$$

If τ is a more complicated type, then it is easiest to think of each relation of τ residing in its own input tape. In this case we use a family of read head predicates. If relation R_j is of arity a, then we split the head positions in R_j into a blocks. The formula $H_t^{j,k}(i)$ will mean that the read head is in relation j at time t and bit i of block k of the head position is one. A bounded number of bits will tell us which relation we are looking at. Suppose we are currently reading a relation R_j of arity a. ONE_t is expressed as follows.

$$\mathrm{ONE}_t \quad \equiv \quad (\exists y_1, \ldots, y_a) \text{ ``}\bar{y} \text{ is the position of the read head at time } t\text{''} \;\; \wedge \;\; R_j(y_1, \ldots, y_a)$$

Where,

$$\text{``}\bar{y} \text{ is the position of the read head at time } t\text{''} \;\equiv\; \bigwedge_{k=1}^{a} (\forall i. B_{\log(n)}(i))(BIT(y_k, i) \leftrightarrow H_t^{j,k}(i))$$

Recall that $a \leq k+1$. If the arity, a, is one, then ONE_t has been written using one domain variable plus a variable restricted to $\log\log n$ variable bits. If $a > 1$, then instead of a new restricted variable i, we can use y_2 to say that y_1 is correct and y_1 to say that the other y's are correct. Thus in either case $VAR[k+1; \log(s(n)/n^k)]$ suffices to express ONE_t.

Next we use the following iterative definition of C_t.

$$C_{t+1}(\bar{x}, \bar{b}) = \bigvee_{\langle h, \bar{a}_{-1}, \bar{a}_0, \bar{a}_1 \rangle \to \bar{b}} \Big(C_t(\bar{x}-1, \bar{a}_{-1}) \wedge C_t(\bar{x}, \bar{a}_0) \wedge C_t(\bar{x}+1, \bar{a}_1) \wedge (\mathrm{ONE}_t \leftrightarrow h) \Big) \quad (3.12)$$

The disjunction runs over the finite set of tuples $(h, \bar{a}_{-1}, \bar{a}_0, \bar{a}_1, \bar{b})$ such that h is a boolean indicating whether the read head is looking at a one and the next three symbols lead to the last symbol in one move of M. This set of tuples is exactly a representation of the state table.

Observe similarly that the next head position relation H_{t+1} can be written in terms of H_t and C_t: we determine as part of the definition of C_{t+1} in Equation 3.12 whether the read head moves to the left or right. Then we must express the fact that H_{t+1} is the successor or predecessor of H_t. In the $\tau \neq \tau_s$ case successor may also involve going from a head position of $n^a - 1$ in one relation to a head position of 0 in the next.

Once we have determined whether H_{t+1} is one more, one less, or equal to H_t, we can express the appropriate condition using two variables restricted to $\log\log n$ bits each.

This completes the proof of Lemma 3.10 and thus of Theorem 3.1. $\square$

4 Conclusions and Directions

The fundamental challenge in computational complexity theory is to understand the trade-off between parallel time and hardware. An exact relationship between quantifier-depth and parallel time on a *CRAM* was previously known (Theorem 2.11). Now we have shown an exact relationship between number of variables and deterministic space. Further work is needed to determine a meaningful, nearly exact relationship between simultaneous descriptive measures and simultaneous parallel time and hardware. This is a tricky problem because it depends on the interconnection patterns and the conventions for concurrent reads and writes. (See [10] for further discussion.)

One instance of the above tradeoff problem seems particularly worthy of study. Consider the following very different characterizations of *PSPACE*.

Theorem 4.1 **([9])**

$$PSPACE \;=\; FO[2^{n^{O[1]}}] \;=\; \mathrm{SO}[n^{0[1]}]$$

Thus, the descriptive power of a bounded number of first-order variables (i.e., $O(\log n)$ bits) and exponential quantifier depth is equal to the descriptive power of a bounded number of second-order variables (i.e., $n^{O[1]}$ bits) and polynomial quantifier depth. We would like to know if there is anything in between. For example, what quantifier depth is necessary and sufficient when $\log n$ first-order variables are available?

To make this problem more concrete, consider the following problem. Define a *k-local graph* to be a graph on vertex set $\{0,1\}^n$ such that for each vertex u there is a unique next vertex v, and the i^{th} bit of v is determined by bits $i-k, i-k+1, \ldots, i+k$ of u. Note that a k-local graph can be presented as a table of size $O[n]$.

Define the local graph reachability problem (LREACH) to be the set of local graphs such that there is a path in the graph from the vertex with all zeros to the vertex with all ones.

The following proposition is clear:

Proposition 4.2 *LREACH is complete for $DSPACE[n]$ via simultaneously logspace and linear time reductions.*

Furthermore, as in Theorem 4.1, the LREACH problem can be expressed in $FO[2^n]\text{-}VAR[2]$ and also as a second-order sentence with three unary relation variables (i.e. $3n$ bits) and quantifier depth $O[n]$.

Now we hope to challenge many people to work hard on the following:

Problem 4.3 *What is the tradeoff between number of variables and quantifier-depth for describing LREACH?*

The main tool currently available for studying Problem 4.3 is the sort of communication complexity game introduced in [13]. A similar game called the separability game is described in [6]. These games as stated only consider circuit depth, and quantifier-depth, respectively. However, one can add a notion of number of variable bits, k, by forcing the players to have only k bits of active memory between rounds, or equivalently, to have at most 2^k different piles to split all the different structures into.

Acknowledgments. Thanks to Sushant Patnaik for asking the question whose answer is the main theorem. Thanks to Bill Hesse for an observation that simplified our proof of Lemma 3.6.

References

[1] D. Mix Barrington, N. Immerman, H. Straubing, "On Uniformity Within NC^1," *JCSS 41, No. 3* (1990) 274 - 306.

[2] J. Cai, M. Fürer, N. Immerman, "An Optimal Lower Bound on the Number of Variables for Graph Identification," *30th IEEE FOCS Symp.* (1989), 612-617.

[3] A. Chandra and D. Harel, "Structure and Complexity of Relational Queries," *JCSS* **25**, 1982, (99-128).

[4] R. Fagin, "Finite-Model Theory – a Personal Perspective," *Third Intl. Conf. Database Theory,* (1990).

[5] Y. Gurevich, "Logic and the Challenge of Computer Science," in *Current Trends in Theoretical Computer Science,* ed. Egon Börger, Computer Science Press (1988), 1-57.

[6] N. Immerman, "Number of Quantifiers is Better than Number of Tape Cells," *JCSS* **22**, No. 3, June 1981, 65-72.

[7] N. Immerman, "Upper and Lower Bounds for First Order Expressibility," *JCSS* **25**, No. 1 (1982), 76-98.

[8] N. Immerman, "Relational Queries Computable in Polynomial Time," *Information and Control,* 68 (1986), 86-104. A preliminary version of this paper appeared in *14th ACM STOC Symp.,* (1982), 147-152.

[9] N. Immerman, "Languages That Capture Complexity Classes," *SIAM J. Comput.* **16**, No. 4 (1987), 760-778.

[10] N. Immerman, *Expressibility and Parallel Complexity, SIAM J. of Comput* **18** (1989), 625-638.

[11] N. Immerman, "DSPACE[n^k] = VAR[$k+1$]," *Sixth IEEE Structure in Complexity Theory Symp.* (July, 1991), 334-340.

[12] N. Immerman, *Descriptive Complexity*, 1998, Springer Graduate Texts in Computer Science, New York.

[13] M. Karchmer and A. Wigderson, "Monotone circuits for connectivity require super-logarithmic depth," *SIAM J. Discrete Math. 3:2* (1990), 255-65.

[14] Y. Moschovakis, *Elementary Induction on Abstract Structures,* North Holland, 1974.

[15] A. Tarksi, "A Lattice-Theoretical Fixpoint Theorem and its Applications," *Pacific. J. Math.*, 55 (1955), 285-309.

[16] M. Vardi, "Complexity of Relational Query Languages," *14th ACM Symposium on Theory of Computing* (1982), 137-146.