INFORMATION AND CONTROL **61**, 65–74 (1984)

A Logic for Constant-Depth Circuits

YURI GUREVICH*

University of Michigan, Ann Arbor, Michigan

AND

HARRY R. LEWIS †

Aiken Computation Laboratory, Harvard University, Cambridge, Massachusetts

Consider a family of boolean circuits $C_1, C_2,\ldots, C_n,\ldots$, constructed by some uniform, effective procedure operating on input n. Such a procedure provides a concise representation of a family of parallel algorithms for computing boolean values. A formula of first-order logic may also be viewed as a concise representation of a family of parallel algorithms for evaluating boolean functions. The parallelism is implicit in the quantification (a formula $\forall x\, \Phi(x)$ is true if and only if each of the formulas $\Phi(a)$ is true, and all these formulas can be checked simultaneously), and universes of different sizes give rise to boolean functions with different numbers of inputs (the boolean values of the formula's predicates on various combinations of elements of the universe). This note presents an extended first-order logic designed to be exactly equivalent in expressiveness to polynomial-size, constant-depth, unbounded-fan-in circuits constructed by Turing machines of bounded computational complexity. © 1984 Academic Press, Inc.

Several papers (Chandra *et al.*, 1983a; Chandra *et al.*, 1982; Chandra *et al.*, 1983b; Furst *et al.*, 1981; Sipser, 1983) have recently dealt with the computational power of boolean circuits with unbounded fan-in to the gates. This model cannot be physically realized but is mathematically natural for many applications and has been studied extensively. To the extent that its computational power can be cleanly characterized in other ways, it is also a "limiting case" for the power of physically realizable limited fan-in circuits (see Sipser, 1983, for a further discussion of the rationale for the model). It is therefore reasonable to seek an alternative formalism which captures exactly the computational strength of such circuits. In this note we describe an extension of first-order logic which can be used to characterize exactly the functions computable by some families of circuits of this type. This proposal

* Supported in part by NSF Grant MCS83-8301022.

† Supported in part by NSF Grants MCS82-03482 and MCS80-05386-A01.

0019-9958/84 $3.00
Copyright © 1984 by Academic Press, Inc.
All rights of reproduction in any form reserved.

is part of a more general program to find logical characterizations of various computational complexity classes (Gurevich, in press; Immerman, 1983).

To understand the general goal, consider the problem of evaluating a fixed first-order formula Φ over an arbitrary finite structure of cardinality n. This can be done by transforming the formula into a circuit: each universal quantifier becomes an n-ary $\wedge$ gate, each existential quantifier an n-ary $\vee$ gate, and the inputs are associated with boolean values of the atomic formulas in the given interpretation. This transformation yields for various n circuits of fixed depth (equal to the maximum syntactic nesting of operators and quantifiers in the formula) and similar structure, which can be constructed in $O(\log n)$ space by a fixed Turing machine on input n (written in unary). However, the circuits so constructed from a first-order formula seem not to form any natural class of computational devices. We therefore seek to enrich the logical language so that the families of circuits constructed by this process form a computationally natural class. That is, any circuit family of this class can be constructed by an effective, uniform process, and it is a consequence of our main result that any circuit family so constructed can be described by a single formula of our extended logic.

For concreteness let S be the set of all mappings computable in space $O(\log n)$, where n is the length of the input. We mention other possibilities for S later. Our logic is parametrized by S and includes, roughly speaking, all predicates in S as primitives. That is, if Φ is a formula of our logic and Φ contains an occurrence of an atomic formula $\mathbf{A}(x_1,..., x_k)$, where $\mathbf{A}$ is the formal name of a predicate A in S, then when Φ is interpreted over the universe $\{0,..., n-1\}$ each instance $\mathbf{A}(m_1,..., m_k)$ of that atomic formula is evaluated by computing the value of A on the integers $m_1,..., m_k$.

To be precise, let k be a fixed integer $\geqslant 0$ and let $A \subseteq N^{k+1}$ be a relation such that any $\langle n, m_1,..., m_k\rangle \in A$ satisfies $n > m_1,..., m_k \geqslant 0$. Then A is said to be *S-computable* if the predicate "$\langle n, m_1,..., m_k\rangle \in A$" is in S, n being written in unary notation. Note that under these circumstances $\langle m_1,..., m_k\rangle$ can be interpreted as a single k-digit, base n numeral representing a number in the range $0,..., n^k - 1$.

We define a logic $L(S)$ which has the following symbols:

Individual variables: x, y, etc.

Logical constants: The symbols $\wedge$, $\vee$, $\sim$, $\forall$, $\exists$. For each $k \geqslant 0$ and for each $(k+1)$-place predicate A in S, a k-place predicate symbol $\mathbf{A}$.

Formulas of L(S) are the usual formulas of first-order logic within this restricted language. They are intepreted over structures with universe $\{0,..., n-1\}$ for some n, and since the interpretations for all symbols are fixed, the truth-value of an L(S) sentence is completely determined by the value of n. We extend the language to include *free predicate variables* as follows:

The logic $\mathrm{L}(S+\mathrm{FPV})$ has the individual variables and the logical constants of $\mathrm{L}(S)$, and, in addition,

Nonlogical symbols: For each $k \geqslant 0$ and each $i \geqslant 0$, a k-place predicate symbol $\mathbf{P}_i^k$.

The formulas of $\mathrm{L}(S+\mathrm{FPV})$ are the formulas of first-order logic within this expanded language.

We define below a class $\mathrm{UC}(S)$ of uniformly constructable families of boolean circuits. The definition resembles Borodin's definition of uniformly constructable circuit families (1977) but makes the complexity of the construction explicit. (When S is of unrestricted complexity, this definition reduces to the nonuniform case.) A Turing machine that constructs a circuit family in $\mathrm{UC}(S)$ can evaluate a predicate A in S on any given tuple of arguments and use the result of this evaluation to direct the construction of a circuit. For this reason these predicates are included as no-cost extensions to the corresponding first-order language $\mathrm{L}(S+\mathrm{FPV})$.

By the *signature* of a formula of $\mathrm{L}(S+\mathrm{FPV})$ we mean the set of its nonlogical predicate symbols (the free predicate variables $\mathbf{P}_i^k$). An $\mathrm{L}(S+\mathrm{FPV})$-formula whose nonlogical predicate symbols are in the signature σ we call an $\mathrm{L}(S+\sigma)$-formula.

Let σ be a signature, i.e., a finite set of nonlogical predicate symbols. Then a σ-structure $\mathbf{M}$ consists of a universe $|\mathbf{M}| = \{0, ..., n-1\}$ for some n, and for each k-place predicate symbol $\mathbf{P}_i^k$ in σ a subset $\mathbf{P}_i^k$ of $|\mathbf{M}|^k$. For example, if σ contains a single binary predicate $\mathbf{P}^2$, then $\mathbf{P}^2$ is the representation of a directed graph. A formula of signature σ is interpreted in a σ-structure of cardinality n in the natural way; in particular each logical constant $\mathbf{A}$ is interpreted as the corresponding predicate A in S with the first component fixed at n. Note that the interpretation of $\mathbf{A}$ in a structure $\mathbf{M}$ depends only on the cardinality of $\mathbf{M}$.

To specify the relation between formulas and circuits we need an equivalence relation on boolean functions. Let f and g be boolean functions of n and m boolean arguments, respectively, and let $p = \max(n, m)$. Then we say that f is *equivalent* to g, in symbols $f \sim g$, provided that for any boolean values $x_1, ..., x_p$, $f(x_1, ..., x_n) = g(x_1, ..., x_m)$. That is, if $n \geqslant m$, then the value of f is independent of its last $n-m$ arguments and f agrees with g when its first m arguments are the same as those of g; and symmetrically if $m \geqslant n$.

Now let σ be a signature with p_i nonlogical predicates of arity d_i. A closed $\mathrm{L}(S+\sigma)$-formula Φ defines, for each n, a boolean function of $N_\Phi(n) = \sum p_i n^{d_i}$ arguments. Specifically, let $F_1, ..., F_{N_\Phi(n)}$ be the list, in lexicographic order, of the $N_\Phi(n)$ atomic formulas $P(m_1, ..., m_j)$, where P is a predicate variable of Φ and $m_1, ..., m_j < n$. Then let Φ_n be the boolean function of $N_\Phi(n)$ arguments such that the value of Φ in a structure $\mathbf{M}$ of cardinality n is the value of $\Phi_n(x_1, ..., x_{N_\Phi(n)})$, where each x_i is the truth value of F_i in **M**. We say that a family $f_1, f_2, \dots,$ of boolean functions is L(S + FPV)-*definable* if there is a formula Φ of L(S + FPV) such that for each $n, f_n \sim \Phi_n$.

EXAMPLES. If $n > 0$, and each of the n^2 numbers $x_{11}, x_{12}, \dots, x_{nn}$ is in $\{0, 1\}$, then the sequence $x_{11}, \dots, x_{nn}$ represents a binary relation R on $\{1, \dots, n\}$, where $R(i,j)$ holds iff $x_{ij} = 1$. Let f_n be that function of n^2 arguments such that $f_n(x_{11}, x_{12}, \dots, x_{nn}) = 1$ if and only if the corresponding relation R on $\{1, \dots, n\}$ is transitive. Then the family $\{f_n\}$ is defined by the L(S + $\{\mathbf{R}\}$)-formula

$$(\forall x)(\forall y)(\forall z)(\mathbf{R}xy \wedge \mathbf{R}yz \Rightarrow \mathbf{R}xz).$$

Let g_n be that function of n^2 arguments such that $g_n(x_{11}, \dots, x_{nn}) = 1$ if and only if the directed graph with vertices $\{1, \dots, n\}$ and with an edge from i to j if $x_{ij} = 1$ is monotone, i.e., has no edges from higher-numbered to lower-numbered vertices. Then $\{g_n\}$ is defined by the $L(S$ + $\{\mathbf{R}\}$)-formula

$$(\forall x)(\forall y)(\mathbf{R}xy \Rightarrow x \leqslant y).$$

Here $\leqslant$ represents a logical constant in S. Finally, let h_n be that function of $n^3 + n^2$ arguments that has the value 1 just in case the last n^2 arguments represent a relation which is the projection of the relation represented by the first n^3 arguments on the first two argument positions. Then h_n is defined by the $L(S$ + $\{\mathbf{R}, \mathbf{S}\}$)-formula

$$(\forall x)(\forall y)(\mathbf{S}xy \Leftrightarrow (\exists z)\, \mathbf{R}xyz).$$

Note that in the examples just given, no logical constant from S is used defining $\{f_n\}$ or $\{h_n\}$; in the example of $\{g_n\}$, S must contain $\leqslant$.

Now we consider a circuit characterization of the L(S + FPV)-definable formulas. Consider a Turing machine which, on input n (in unary), produces via a mapping in S a circuit C_n of polynomial-bounded size, depth bounded by a constant independent of n, and unbounded fan-in. (The *size* of a circuit is the number of edges. Alternatively, the size of a circuit is the total size of all its gates, where the size of a gate is its fan-in. Since our circuits have polynomial-bounded size, both the total number of gates and the fan-in at each gate are polynomial bounded. To be precise, the size of a circuit is at most the product of the number of gates and the maximum fan-in. The *depth* of a node is the length of the longest path from that node to the root, and the *depth* of the circuit is the maximum depth of any node.) The complexity class S should be sufficiently powerful that this notion of circuit construction is robust and independent of details of the encoding, but for concreteness we give a specific convention. Let p be a polynomial such that the circuit produced on input n has at most $p(n)$ gates. Also, let $I_n \leqslant p(n)$ be the number of inputs of circuit C_n. Without loss of generality we may assume that the Turing machine first enumerates the gates as numbers 1, 2, etc., up to at most $p(n)$, each with its associated boolean function ($\wedge$, $\vee$, or NOT) or a designation that it is an INPUT gate. We assume that the input gates are numbered $1, 2, \ldots, I_n$. Since $p(n)$ is a polynomial of fixed degree, the gate numbers can be written in base n notation as sequences of fixed length. The Turing machine then lists the edges as ordered pairs of gate numbers. We assume that the circuits are acyclic and have a distinguished output—the root. Thus such a Turing machine defines, for each n, a boolean function of exactly I_n boolean values, to wit, the values of the input gates $1, 2, \ldots, I_n$. A collection of boolean functions definable in this way we call *uniformly S-computable* or UC(S).

Note that if S contains all log space computable mappings, then the uniformly S-computable functions are in fact uniformly S-computable by means of polynomial size, constant depth circuits that are "almost trees" in the sense that their noninput nodes form trees. (Therefore readers who prefer to think of trees rather than general circuits are free to do so.) To see this, suppose the circuit produced on input n has size at most $p(n)$, and therefore has at most $p(n)$ gates and $p(n)$ edges, and is not an "almost tree." Let d be the smallest depth such that there is at least one noninput node of depth d with fan-out greater than 1. Replace each noninput node N at depth d having fan-out $b > 1$ by b copies of itself, each having fan-out 1 and having fan-in from the same nodes as did N. Clearly the resulting circuit computes the same function and has no noninput nodes with fan-out greater than 1 at depth less than $d+1$. Moreover the size of the resulting circuit is at most $p(n)^2$. This construction can then be iterated for nodes of greater depth until no noninput nodes remain that have fan-out greater than 1. If the depth of the original circuit was D (independent of n) then the final circuit has depth D and size at most $p(n)^{2^D}$, which is polynomial in n. Moreover this construction can be carried out in log space.

Before stating and proving the main result it is worth pointing out that the UC(S) families of functions are identical to a class of function families computed by certain circuits with *bounded* fan-in. By DeMorgan's laws we may assume that any occurrence of a NOT gate in a circuit is the parent of an INPUT gate; such a circuit we call *normalized.* An *alternation* in a circuit is an edge between an AND gate an an OR gate (the AND gate is the parent of the OR gate or vice versa). The *alternation depth* of a circuit is the maximum number of alternations on any path from the root to an input. Then a collection of boolean functions is UC(S) if and only if it is the set of functions computed by a collection of normalized circuits C_1, C_2,... constructed by a Turing machine of complexity S such that C_n, the circuit constructed on input n, is of fan-in bounded by a constant, depth $O(\log n)$, and alternation depth bounded by a constant. For any subcircuit not containing an alternation can be collapsed into a single gate of polynomial-bounded fan-in, and the unbounded fan-in circuit that results from a maximal application of this process has depth equal to the (bounded) alternation depth of the original circuit. Conversely, a polynomial-bounded fan-in gate can be expanded into $O(\log n)$ levels of gates each of constant fan-in.

THEOREM. *Suppose that S contains all log space computable mappings and is closed under composition. Then a family F of boolean functions is* $\mathrm{L}(S + \mathrm{FPV})$*-definable if and only if it is* $\mathrm{UC}(S)$.

Proof. (I) *IF F is* $\mathrm{L}(S + \mathrm{FPV})$*-definable, then F is* $\mathrm{UC}(S)$. Given a formula of $\mathrm{L}(S + \mathrm{FPV})$, we can construct a Turing machine which on input n generates the naturally corresponding circuit, with $\wedge$ and $\forall$ being replaced by $\wedge$-gates, $\vee$ and $\exists$ being replaced by $\vee$-gates, $\sim$ by $\sim$-gates, and occurrences of atomic formulas $\mathbf{A}(m_1,\ldots, m_k)$, where $\mathbf{A}$ is a logical constant, by the constants 0 and 1 (these predicates can be evaluated within the given complexity bound). The expansion of the formula as a circuit proceeds recursively in depth-first fashion, so that at any time at most one binding for each variable of the formula need be remembered. The resulting circuit is a tree except at the bottom level, where some leaves are connected to several nodes at levels above the bottom. At the bottom level, atomic formulas $\mathbf{P}_i^k m_1 \cdots m_k$ become input nodes of the circuit. Different truth valuations of these atomic formulas correspond to different assignments of boolean values to the circuit inputs.

For example, Fig 1 shows the circuit that would be generated for the formula $\forall y\, \exists x\, (\mathbf{P}xy \wedge \mathbf{P}yx \wedge \mathbf{A}x)$ in case $n = 2$ and $A0 = 1$, $A1 = 0$. This circuit has 15 gates. The four input gates correspond to the values of the atomic formulas P00, P01, P10, P11; there are five "and" and two "or" gates; and there are four gates with constant values (0 or 1).

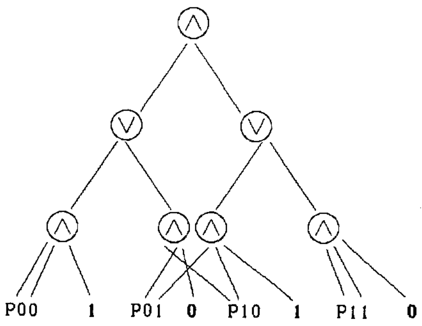

FIGURE 1

(II) *If F is* UC(S), *then F is* L(S + FPV)*-definable.* Suppose that Turing machine M operates within the complexity bound S and, for some polynomial p and some constant d, produces on input n a circuit C_n of depth d with I_n inputs and with at most $p(n)$ gates. Without significant loss of generality let us assume that for some constant e, $p(n) = n^e$. (This assumption enables us to construct a formula with a single e-place predicate letter **P**. If p is not bounded by a monic polynomial, i.e., if $p(1) > 1$, then several predicate letters need to be used in the construction. If the $n = 1$ case is simply ignored, this assumption entails no loss of generality.) Thus each gate g listed by M can be named by a sequence of e integers in the range $0, \dots, n-1$. We call this the *internal* name of M for g.

Moreover, there is a constant c such that, for each n, each gate of the circuit produced on input n has fan-in at most n^c. Thus each gate listed by M can also be identified with one or more sequences integers, each in the range $0, \dots, n-1$, by following paths from the root to the gate. If the path followed has length i, then the identifying sequence is of length ci; we call such an identifying sequence an *address* of the gate in the circuit. The address of the root is the empty sequence $\langle\,\rangle$ and, if gate g has address $\langle m_1, \dots, m_{ci}\rangle$ by a path of length i, and gate g' is the k'th child of g, where $k = q_1 n^{c-1} + q_2 n^{c-2} + \cdots + q_c$ $(0 \leqslant q_1, \dots, q_c \leqslant n-1)$, then gate g' has address $\langle m_1, \dots, m_{ci}, q_1, \dots, q_c\rangle$. The same gate may have several addresses since there may be several different paths from the root to a gate. Not every sequence $\langle m_1, \dots, m_{ci}\rangle$ need be the address of a gate. The only gates with 0 children are input gates and gates with the constant value 0 or 1.

We claim that for each $i = 0, \dots, d$, the following predicates are S-computable:

Address$_i(n, m_1, \dots, m_{ci})$: $\langle m_1, \dots, m_{ci}\rangle$ is the address of a gate

And$_i(n, m_1, \dots, m_{ci})$: There is a gate in C_n with address $\langle m_1, \dots, m_{ci}\rangle$ which is an "and" gate.

Or$_i(n, m_1, \dots, m_{ci})$: There is a gate in C_n with address $\langle m_1, \dots, m_{ci}\rangle$ which is an "or" gate.

Not$_i(n, m_1, \dots, m_{ci})$: There is a gate in C_n with address $\langle m_1, \dots, m_{ci}\rangle$ which is a "not" gate.

Input$_i(n, m_1, \dots, m_{ci})$: There is a gate in C_n with address $\langle m_1, \dots, m_{ci}\rangle$ which is an input gate.

One$_i(n, m_1, \dots, m_{ci})$: There is a gate in C_n with address $\langle m_1, \dots, m_{ci}\rangle$ which has the constant value 1.

Equiv$_i(n, m_1, \dots, m_{ci}, p_1, \dots, p_e)$: There is a gate in C_n with address $\langle m_1, \dots, m_{ci}\rangle$ and with internal name $\langle p_1, \dots, p_e\rangle$.

To see that these are all S-computable predicates, note that, given our standard presentation of C_n, each of them can be computed in the process of constructing C_n itself by using $O(\log n)$ space. Also, note that if there is a gate in C_n with address $\langle m_1,..., m_{ci}\rangle$ which is an "and," "or," or "not" gate, then there is also at least one gate in C_n with address of length $c(i+1)$ and beginning with $m_1,..., m_{ci}$, i.e., of the form $\langle m_1,..., m_{ci}, m_{ci+1},..., m_{c(i+1)}\rangle$ for some $m_{ci+1},..., m_{c(i+1)}$.

We now construct a formula Φ_0 of the logic L(S + FPV) with one uninterpreted e-place predicate letter **P** such that the value of Φ under an interpretation with universe $\{0,..., n-1\}$ is the same as the value of the circuit C_n, when the atomic formula $\mathbf{P}p_1 \cdots p_e$ has the same truth-value as the input gate of C_n, if any, with internal name $\langle p_1,..., p_e\rangle$. (As mentioned above, if the number of input gates of C_n is not a monic polynomial in n, then more than one nonlogical predicate **P** would have to be used in the construction of Φ_0.) Φ_0 is the last of a sequence $\Phi_d,..., \Phi_0$, where Φ_i has ci free variables and gives the value of a gate with an address of length i in terms of the values of gates with longer addresses.

$$\begin{aligned}
&\Phi_d(x_1,..., x_{cd}) \text{ is}\\
&One_d(x_1,..., x_{cd}) \vee\\
&\quad \exists y_1 \cdots y_e(Input_d(x_1,..., x_{cd}) \wedge\\
&\qquad\qquad Equiv_d(x_1,..., x_{cd}, y_1,..., y_e) \wedge \mathbf{P}y_1 \cdots y_e);
\end{aligned}$$

and for $i < d\Phi_i(x_1,..., x_{ci})$ is

$$\begin{aligned}
&One_i(x_1,..., x_{ci}) \vee\\
&\quad (\exists y_1 \cdots y_e(Input_i(x_1,..., x_{ci}) \wedge\\
&\qquad\qquad Equiv_i(x_1,..., x_{ci}, y_1,..., y_e) \wedge \mathbf{P}y_1 \cdots y_e)\\
&\quad \vee (And_i(x_1,..., x_{ci}) \wedge\\
&\qquad \forall x_{ci+1} \cdots x_{c(i+1)}(Address_{i+1}(x_1,..., x_{c(i+1)}) \Rightarrow \Phi_{i+1}(x_1,..., x_{c(i+1)})))\\
&\quad \vee(Or_i(x_1,..., x_{ci}) \wedge\\
&\qquad \exists x_{ci+1} \cdots x_{c(i+1)}(Address_{i+1}(x_1,..., x_{c(i+1)}) \wedge \Phi_{i+1}(x_1,..., x_{c(i+1)})))\\
&\quad \vee (Not_i(x_1,..., x_{ci}) \wedge \sim\Phi_{i+1}(x_1,..., x_{ci}, 0,..., 0)).
\end{aligned}$$

This completes the construction. At stage i of the construction, quantification over c individual variables $x_{ci+1},..., x_{c(i+1)}$ serves to represent the fan-in of a gate with an address of length i from at least one, but at most n^c, gates with addresses of length $i+1$. The boolean function to be computed at that gate is determined by the predicates And_i, Or_i, and Not_i and is then rendered in the logical formula by restricted universal or existential quantification or by negation. Quantification is restricted by the condition $Address_{i+1}(x_1,..., x_{c(i+1)})$ to child gates that actually exist. The fact that Φ_d, $\Phi_{d-1},..., \Phi_0$ have the correct values follows by induction, using the previously stated semantics for the predicates $Address_i$, And_i, Or_i, Not_i, One_i, and $Equiv_i$. To be precise, $N_{\Phi_0}(n) = n^e$, and Φ_0 is a boolean function of the n^e boolean values for the atomic formulas $\mathbf{P}p_1 \cdots p_e$, where $0 \leqslant p_1,..., p_e < n$, and more specifically of the values of the first I_n atomic formulas $\mathbf{P}p_1 \cdots p_e$ (those such that for some i, $m_1,\dots, m_{ci}$, both $Input_i(n, m_1,\dots, m_{ci})$ and $Equiv_i(n, m_1,\dots, m_{ci}, p_1,\dots, p_e)$ hold). The size of the formula is independent of n and is determined by the constants c, d, and e which give the degrees of the polynomials limiting the fan-in, depth, and number of gates of the circuit. It is therefore polynomially related to the size of the circuit, which is $O(n^{ce})$. ∎

The proof of the theorem requires of S somewhat less than is stated in the hypothesis. All that is really required is that the composition of any logspace computable mapping with any mapping in S is also in S, so that in the proof of (II) the application of a logspace mapping to the construction of C_n is a mapping in S. The theorem and its proof are valid for many classes of mappings, e.g., for the classes of mappings computable in polynomial time, or polynomial space, or $O(\log^2 n)$ space, or polylog space, etc.

Also, we may take S to be the set of all mappings, computable or otherwise. Then UC(S) contains all nonuniform families of polynomial-size, constant-depth circuits. In other words, let C_1, $C_2,\dots$, be any sequence of polynomial-size, constant-depth circuits. Say the size of C_n is $O(n^k)$ and its depth is d, independent of n. Then the construction described above yields a formula Φ_0 of L(S + FPV) containing predicates $Equiv_i$, And_i, etc., which are in general uncomputable. The size of Φ_0 depends on k and d only. Compare in this connection the note on nonuniform circuit complexity in Immerman (1983).

The theorem also provides a characterization, albeit a fairly complex one, of L(S), the logic without free predicate variables and with logical contants for the S-computable mappings only. However, in several cases L(S) can be presented in a more elegant way. As a concrete example let us return to the case of S = the logspace computable mappings. According to Immerman (1983), L(S) can be described as the extension of the first-order theory of linear order by means of the so-called deterministic transitive closure operation.

Another way to extend the language of first-order logic to define the log space computable predicates is to allow definition of functions by the primitive recursion schema. Gurevich (1983) shows that if numbers up to n^k are presented by k-tuples of digits less than n, then the logspace computable functions are exactly those obtainable from a few base functions by composition and the schema

$$f(\mathbf{x}, \mathbf{0}) = g(\mathbf{x})$$
$$f(\mathbf{x}, \mathbf{t}+1) = h(\mathbf{x}, \mathbf{t}, f(\mathbf{x}, \mathbf{t})).$$

Either characterization of L(S) obviously gives rise to a more elegant version of L(S + FPV).

One final remark. L(LOGSPACE + FPV) is a relatively modest extension of first-order logic which is computationally natural. In this connection it would be interesting to extend first-order logic as little as possible while still obtaining a computationally natural system.

ACKNOWLEDGMENTS

We thank the referee for a correction to a previous version of this paper. We are grateful to S. Tucker Taft for assistance with the preparation of the manuscript.

RECEIVED: August 22, 1983; ACCEPTED: July 1984

REFERENCES

BORODIN, A. (1977), On relating time and space to size and width, *SIAM J. Comput.* **6**, 733–744.

CHANDRA, A. K., FORTUNE, S., AND LIPTON, R. (1983a), Unbounded fan-in circuits and associative functions, *in* "Proc. 15th Annual ACM Sympos. Theory of Comput.," pp. 52–60.

CHANDRA, A. K., STOCKMEYER, L. J., AND VISHKIN, U. (1982), A complexity theory for unbounded fan-in parallelism, *in* "Proc. 23rd Annual IEEE Sympos. Found. Comput. Sci.," pp. 1–13.

CHANDRA, A. K., STOCKMEYER, L. J., AND VISHKIN, U. (1983b), "Constant Depth Reducibility," IBM Technical Report.

FURST, M., SAXE, J. B., AND SIPSER, M. (1981), Parity, circuits, and the polynomial-time hierarchy, *in* "Proc. 22nd Annual IEEE Sympos. Found. Comput. Sci.," pp. 260–270.

GUREVICH, Y. (in press), Toward a logic tailored for computational complexity, *in* "Proc. 1983 European Logic Colloquium," Springer-Verlag, Berlin/New York.

GUREVICH, Y. (1983), Algebras of feasible functions, *in* "Proc. 24th Annual IEEE Sympos. Found. Comput. Sci." pp. 210–214.

IMMERMAN, N. (1983), Languages which capture complexity classes, *in* "Proc. 15th Annual ACM Sympos. Theory of Computing," pp. 347–354.

SIPSER, M. (1983), Borel sets and circuit complexity, *in* "Proc. 15th Annual ACM Sympos. Theory of Computing," pp. 61–69.