Bounded-Width Polynomial-Size Branching Programs Recognize Exactly Those Languages in NC^1

David A. Barrington[1]
Department of Mathematics
Massachusetts Institute of Technology
Cambridge, MA 02139

1. Abstract

We show that any language recognized by an NC^1 circuit (fan-in 2, depth $O(\log n)$) can be recognized by a particular type of width 5 polynomial-size branching program. As any bounded-width polynomial-size branching program can be simulated by an NC^1 circuit, we have that the class of languages recognized by such programs is exactly non-uniform NC^1. Further, following Ruzzo [Ru81] and Cook [Co85], if the branching programs are restricted to be $ATIME(\log n)$ uniform, they recognize the same languages as do $ATIME(\log n)$-uniform NC^1 circuits, that is, those languages in $ATIME(\log n)$. We also extend the method of proof to investigate the complexity of the word problem for a fixed permutation group.

2. Definitions

We define a branching program of width w (a w-BP) to be a sequence of instructions $\langle j_i, f_i, g_i \rangle$ for $1 \leq i \leq l$, where x_{j_i} is one of n input variables, f_i and g_i are functions from $[w]$ to $[w]$ (throughout, $[w]$ is the set $\{0, \ldots, w-1\}$), and l is the length. Given a setting of the input variables, the instruction $\langle j_i, f_i, g_i \rangle$ yields the function f_i if x_{j_i} is on and g_i if x_{j_i} is off. A branching program B on input setting $\mathbf{x}$ yields the composition of the functions yielded by each of its instructions – we call this composition $B(\mathbf{x})$. B is a permutation branching program (w-PBP) if each f_i and g_i is a permutation of $[w]$ – in general we will use Greek letters for permutations.

The traditional notion (e.g., in [BDFP83]) of a width-w branching program is of a rectangular w by l array of nodes, each assigned an input variable and two edges – one to be followed if the variable is on and the other if it is off. Our model can be viewed this way but further requires that the branching program be leveled (an edge must terminate at a node in the next column to the right of its source) and that each node in a column be assigned the same variable. These requirements may be enforced at a cost of increasing the length polynomially and doubling the width. Thus both models define the same class $BWBP$ of languages recognized by bounded-width polynomial-size branching programs. Our main result is that $BWBP = (\text{non-uniform})NC^1$.

In general we will speak of non-uniform circuits and branching programs, so that an NC^1 circuit will be simply a Boolean circuit with fan-in 2 and depth $O(\log n)$ (and hence polynomial size). Uniformity considerations will be postponed until Section 6, where we will make the appropriate definitions.

[1]This work is supported by NSF grant MCS-8304769, US Air Force grant AFOSR-82-0326, and by an NSF Graduate Fellowship.

3. Previous Work

Branching programs were defined by Lee [Le59] as an alternative to Boolean circuits in the description of switching problems – he called them 'binary decision programs'. They were later studied in the Master's thesis of Masek [Ma76] under the name of 'decision graphs'.

Borodin, Dolev, Fich and Paul [BDFP83] raised the question of the power of bounded-width branching programs. They noted that the class BWBP contains AC^0 (languages recognized by unbounded fan-in, constant-depth, polynomial-size Boolean circuits) as well as the parity function (shown to be outside AC^0 in [FSS81] and [Aj83]). They conjectured that the majority function was not in $BWBP$, in fact that for bounded width it requires exponential length.

Subsequent results appeared to support this conjecture. Chandra, Furst, and Lipton [CFL83] and Púdlak [Pu84] showed linear and superlinear length lower bounds respectively for arbitrary constant width. In [BDFP83] the idea was to work with width-2 and get exponential bounds. They succeeded for a restricted class of BP's , and Yao [Ya83] followed with a superpolynomial lower bound for general width-2. Shearer [Sh85] proved an exponential lower bound for the mod-3 function with general width-2. Ajtai et al. have just proved a nearly $n \log n$ size lower bound for a large class of symmetric functions [ABHKST86], where width is unconstrained but size is defined to be length times width.

Barrington [Ba85] revised the notion of width to the one used here and considered width-3 permutation branching programs. Their power was characterized as equal to that of certain depth-2 circuits of mod-2 and mod-3 gates, and it was shown that these could recognize any set in exponential length and that exponential length was required to recognize a singleton set.

Here we show that since the majority function is in NC^1, it is in $BWBP$, and thus the conjecture of [BDFP83] is false. While polynomial-size 3-PBP's have very limited power, we show that polynomial-size 5-PBP's suffice to simulate NC^1 circuits.

4. The Main Result

We say that a 5-PBP B five-cycle recognizes a set $A \subseteq [2]^n$ if there exists a five-cycle σ (called the output) in the permutation group S_5 such that $B(\mathbf{x}) = \sigma$ if $\mathbf{x} \in A$ and $B(\mathbf{x}) = e$ if $\mathbf{x} \notin A$ (e is the identity permutation).

Theorem 1: Let A be recognized by a depth d fan-in 2 Boolean circuit. Then A is five-cycle recognized by a 5-PBP B of length at most 4^d.

Lemma 1: If B five-cycle recognizes A with output σ and τ is any five-cycle, then there exists a 5-PBP B', of the same length as B, which five-cycle recognizes A with output τ.

Proof: Since σ and τ are both five-cycles there exists some permutation θ with $\tau = \theta\sigma\theta^{-1}$. To get B', simply change each instruction of B, replacing each σ_i and τ_i by $\theta\sigma_i\theta^{-1}$ and $\theta\tau_i\theta^{-1}$.

Lemma 2: If A is five-cycle recognized in length l, so is its complement.

Proof: Let B five-cycle recognize A with output σ. Call the last instruction of B $\langle i, \mu, \nu\rangle$. Let B' be identical to B except for last instruction $\langle i, \mu\sigma^{-1}, \nu\sigma^{-1}\rangle$. Then $B'(\mathbf{x}) = e$ if $\mathbf{x} \in A$ and $B'(\mathbf{x}) = \sigma^{-1}$ if $\mathbf{x} \notin A$. Thus B' five-cycle recognizes the complement of A.

Lemma 3: There are two five-cycles σ_1 and σ_2 in S_5 whose commutator is a five-cycle. (The commutator of a and b is $aba^{-1}b^{-1}$.)

Proof: $(12345)(13542)(54321)(24531) = (13254)$.

Proof of Theorem: By induction on d. If $d = 0$ the circuit is an input gate, and A can easily be recognized by a one-instruction 5-PBP. Using Lemma 2 in the case of an or-gate, assume without loss of generality that $A = A_1 \cap A_2$, where A_1 and A_2 have circuits of depth $d-1$ and thus 5-PBP's B_1 and B_2 of length at most 4^{d-1}. Let B_1 and B_2 have outputs σ_1 and σ_2 as in Lemma 3, and B_1' and B_2' have outputs σ_1^{-1} and σ_1^{-1} (This last is possible by Lemma 1). Let B be the concatenation $B_1B_2B_1'B_2'$. B yields e unless the input is in both A_1 and A_2, but yields the commutator of the two outputs if the input is in A. This commutator is a five-cycle, and so B five-cycle recognizes A. B has length at most 4^d. Given a circuit and a desired output, this proof gives a deterministic method of constructing the 5-PBP.

The following result is a non-uniform version of the well-known result that regular languages can be recognized by NC^1 circuits. The proof in the uniform case essentially appears in [Sa72] and is given explicitly in [LF77].

Theorem 2: If $A \subseteq [2]^n$ is recognized by a w-BP B of length l, A is recognized by a fan-in 2 circuit of depth $O(\log l)$ where the constant depends on w.

Proof: We may choose the weakest possible notion of recognition here and say that B accepts $\mathbf{x}$ if $B(\mathbf{x})$ is in some arbitrary subset of the functions from $[w]$ to $[w]$. We can represent such a function f by w^2 Boolean variables telling whether $f(i) = j$ for each i and j. The composition of two such functions so represented may be computed by a fixed circuit whose size depends only on w. Our circuit for A will have a constant-depth section to find the function yielded by each instruction of B, a binary tree of composition circuits, and a constant-depth section at the top to determine acceptance given the function yielded by B.

Corollary: The classes of languages $BWBP$ and non-uniform NC^1 are identical. They equal the class of languages recognized by polynomial-length 5-PBP's.

5. Some Consequences

The intuition that the width of a branching program corresponds to the number of possible configurations of an automaton appears to be wrong. We can formalize this notion to some extent. Define a non-uniform DFA (NUDFA) to be a k-state automaton with a two-way, read-only input tape and a one-way program tape. The latter contains instructions to move right or left on the input or to change state depending on the current state and the visible input character. NUDFA's with k states and k-BP's simulate each other – BP length corresponds to program tape length up to a multiplicative factor of $O(n)$. NUNFA's can be similarly defined, and the familiar subset construction can be carried out showing that they have the same power (for a given length) as NUDFA's (a 2^k-state NUDFA can simulate a k-state NUNFA). This shows, given the corresponding definition of nondeterministic branching programs that nondeterministic $BWBP$ is also non-uniform NC^1. Our intuition must accept the fact that a 5-state NUDFA can count in polynomial time, as all symmetric functions are in NC^1. It can also divide, as by [BCH84] integer division is in non-uniform NC^1.

If we define the width of a Boolean circuit in such a way as not to charge for wires from any node to an input, then NC^1 languages may all be recognized by polynomial-size circuits of constant width. This follows immediately from the main result.

6. Uniformity:

We define an alternating Turing machine to be a game played by two players on a nondeterministic Turing machine which has two possible state transitions in every position. States are labelled White or Black as to which player has control of the moves from that state. For defining the class $ATIME(\log n)$ we assume that the machine has a random-access input tape which it can access only once, at the end of the computation, a worktape of size $c \log n$ for some constant c, and a clock which restricts it to running for $c \log n$ steps. The players, who are assumed to be omniscient, direct the computation of the machine until the end, when White wins iff he can correctly predict the

input bit to be read. The alternating Turing machine is said to accept an input $\mathbf{x}$ if White has a winning strategy for this game with input $\mathbf{x}$. By standard methods these assumptions may be shown to be perfectly general.

Ruzzo [Ru81] defines NC^1 circuits as those fan-in 2 depth $O(\log n)$ circuits whose extended connection language is in the class $ATIME(\log n)$. The extended connection language consists of strings of the form $\langle g, h, s\rangle$ where g and h are names of nodes in the circuit, $s \in \{left, right\}^{\leq \log n}$, and h is the node reached by following the path s from g. This has the consequence that $NC^1 = ATIME(\log n)$. We would like to show that the class of languages recognized by $ATIME(\log n)$-uniform polynomial-size bounded-width branching programs is also $ATIME(\log n)$. This will show that $BWBP = NC^1$ in the uniform as well as in the non-uniform setting.

Theorem 3: A language A is in $ATIME(\log n)$ iff it is recognized by a polynomial-size bounded-width branching program B for which the language:

$$\{\langle k, f, g, i\rangle : \text{the } k\text{'th instruction of } B \text{ yields function } f \text{ if } x_i \text{ is on and } g \text{ if } x_i \text{ is off}\}$$

is in $ATIME(\log n)$.

Proof: First we define a game in which White tries to prove that $B(\mathbf{x}) = f$, for some accepting f, and Black tries to refute him. At each stage of the game the log-time machine will define a range of instructions in B and a function which White claims is yielded by that range. White advances his claim by naming two functions g and h, with $f = gh$, and claiming that the first half of the range yields g and the second h. Black must choose one of these two subclaims to challenge, and this becomes White's new claim for the next stage. After $O(\log n)$ stages White will be making a claim about a single instruction, and this can be verified in $ATIME(\log n)$ by hypothesis. Each stage takes constant time, as we can let Black's sequence of choices be the index of the instruction to be checked – so each bit of this index need only be written down once.

For the converse, given a log-time machine M and game rules to make it an alternating machine, we can get an NC^1 circuit C in a standard way by creating a node for each configuration of M. Let B be the 5-PBP with output (12345), say, created from C by the method of Theorem 1 above, so that B five-cycle recognizes A. We must show that B is $ATIME(\log n)$ uniform. We define a game with input $\langle k, \sigma, \tau, i\rangle$ which White can win iff the input is a correct description of the k'th instruction. Both players, of course, know the actual circuit C and branching program B, as these are uniquely defined from M.

White at each stage will maintain a claim of the following form:

$$\langle s, \mu, k, \sigma, \tau, i\rangle$$

meaning 'The subcircuit C_s of C, whose top node is the configuration s of M, corresponds to a section B_s of B which five-cycle recognizes the language accepted by C_s with output μ. Further, the k'th instruction of B_s yields σ if x_i is on and τ if x_i is off.'

White will begin by claiming $\langle start, (12345), k, \sigma, \tau, i\rangle$ and refine this through $O(\log n)$ moves, each move corresponding to a step of M or to moving down one edge of C. For example, if s is an and-node B_s consists of four sections – White must state in which section the k'th instruction occurs, what its new number is, and which of s's children the section represents. Eventually s will be a final configuration of M and White's claim can be quickly decided. Black's moves during this process are to challenge any White claim which does not follow from his previous claim according to the definition of M and the procedure for creating B. Such a challenge may be decided easily in log-time ending the game. White's moves are each only a constant number of steps if we choose an appropriate representation for the number k and don't have to rewrite it every time.

7. Extensions and the Fine Structure of NC^1

What we have really done in Theorem 1 is to show that a certain problem is complete for NC^1 under certain reductions. The problem is to multiply together a series of elements of S_5, or equivalently to test whether a given word over S_5 is the identity. We will call this the *word problem* for S_5. Similarly we may define the word problem for any fixed group G. (We consider only finite groups, and always assume a group is represented as a permutation group.) This will correspond to the class of G-PBP's, where the permutations in each instruction must belong to G. Thus, for example, w-PBP's become S_w-PBP's.

Inside NC^1, it is most natural to define AC^0 reductions – the function f is reducible to g (written $f \leq_{AC^0} g$) if a constant-depth poly-size unbounded fan-in circuit, containing oracle nodes for g, can compute f. This notion was introduced in [FSS81] under the name of 'cp-reducibility'. They suggested further study of the degree structure (they had only just given the first proof that the structure of NC^1 was non-trivial) and conjected that majority was not reducible to parity.

This study was taken up by Fagin et al. in [FKPS84], who found many new AC^0 reducibilities among symmetric functions. Modulo the new parity lower bounds of Yao [Ya85] and Hastad [Ha86], they characterize those symmetric functions in AC^0. They show that the degree of the majority function is complete for symmetric functions and contains a large class of symmetric functions. Interestingly, no complete symmetric function exists in the projection-reducibility theory of Valiant [SV81], by a recent result of Geréb-Graus and Szemerédi [GS??].

In this section we show that *solvability* of a group is the key to the applicability of the methods used earlier for the group S_5. We first give one of the many equivalent definitions (For more detail see a group theory text such as [Za58]). The commutator subgroup of G is the subgroup generated by all elements of the form $aba^{-1}b^{-1}$ for a and b in G. A group is solvable if and only if repeated taking of commutator subgroups eventually gives the trivial group. Thus a group is non-solvable if and only if it has a nontrivial subgroup whose commutator subgroup is itself. (All groups under discussion are finite.)

We first show that our earlier proof generalizes to any non-solvable group.

Theorem 4: The word problem for any fixed non-solvable group G is complete for NC^1 under AC^0 reductions.

Proof: Without loss of generality, assume that G's commutator subgroup is itself. We show that given a fan-in 2 circuit of depth d and an element a of G not equal to the identity, there is a G-PBP of length at most $(4g)^d$ which yields a if the circuit

accepts the input and yields the identity otherwise. Here g is the order of G, a constant. Evaluating a G-PBP is easily seen to be in AC^0, given oracle nodes for the word problem for G. This will suffice to show completeness – the word problem is clearly in NC^1 as we can multiply two permutations in constant size and depth with fan-in two.

The proof, like that of Theorem 1, is by induction on d. The element a must have a representation as a product of at most g commutators. We carry out the proof of Theorem 1, except that we use the inductive hypothesis to produce G-PBP's yielding arbitrary non-identity elements of G instead of five-cycles. This multiplies the length by at most $4g$ instead of 4 at each step. Lemma 1 is unnecessary as for each d, we simultaneously prove the result for all a in G except the identity.

It would be nice to have a converse to Theorem 4, but unfortunately we do not know enough about the AC^0-structure of NC^1 to prove one. By extending the methods of [Ba85], however, we can make a good start.

Theorem 5: The word problem for any fixed solvable group G is AC^0-reducible to the mod g function, where g is the order of G.

Proof: An equivalent definition of a solvable group (see, e.g., [Za58]) is one which has a series of normal subgroups $G = G_0, G_1, \ldots, G_m = \{e\}$ where each quotient group G_i/G_{i+1} is cyclic. We prove the theorem by induction on the length of this series. So assume that G has a normal subgroup N, where G/N is cyclic and the word problem for N is solvable by an AC^0 circuit containing mod g gates. Choose an element a such that the coset aN generates G/N.

We are given a product $g_1 \ldots g_k$ to evaluate. As N is normal, we can write each g_i uniquely as $a^{\epsilon_i} n_i$ with $n_i \in N$. (Converting between any two bit representations of an element of G takes constant size and depth.) Now let b_i be the product $a^{\epsilon_1} \ldots a^{\epsilon_i}$ and note that:

$$a^{\epsilon_1} n_1 \ldots a^{\epsilon_k} n_k = (b_1 n_1 b_1^{-1}) \ldots (b_k n_k b_k^{-1}) b_k.$$

Each b_i depends only on the sum mod g of the appropriate ϵ_j, as the order of a in G divides g. Each term $b_i n_i b_i^{-1}$ is in N by normality, and we can calculate it in constant depth using mod g gates to get b_i. These partial terms may then be multiplied using a circuit for N.

Theorem 5 is interesting only if the converse of Theorem 4 is true. To finish the argument, we would need to show that no single mod g function is powerful enough to be complete for NC^1. We conjecture that this is true, as it seems quite unlikely that it could do majority. This extends the conjecture of [FSS81] that the mod 2 function is not powerful enough to do majority. Unfortunately, the random restriction method of [FSS81] does not seem to extend to even parity (mod 2) gates, as the restriction of a parity gate is still a parity gate. It appears that an entirely new technique is needed.

8. Open Problems

We now know that poly-size bounded-width BP's give NC^1 while poly-size general BP's give L. Certainly this suggests a new attack on the problem of whether $NC^1 = L$ as this can now be phrased entirely in terms of branching programs. It would be useful to develop a lower-bound technology for width-5 PBP's if this is possible. Even a superpolynomial lower bound for, say the clique function would give prove NC^1 different from NP.

We don't know the power of general poly-size permutation BP's (no restriction on width). They might recognize NC^1 or L (the former seems quite unlikely to me) or some class in between. These PBP's can be thought of as reversible non-uniform log-space Turing machine computations – this suggests a comparison with work of Bennett [Be73]. In the language of Section 7, we can ask whether the word problem for S_n is complete for L if input and output are given in list notation rather than cycle notation (i.e., a permutation σ is given as the list of integers $\sigma(1), \ldots, \sigma(n)$). Similar problems are among the list of L-complete problems given by Cook [Co85]. Small changes in the statement of such problems can be important, as for example changing a permutation in S_n from list to cycle notation is itself L-complete. The study of poly-size PBP's is also sensitive to changes in the definition of recognition of a language.

The effect of non-determinism on these classes must be examined as well, suggesting possible new attacks on the problem of whether $L = NL$. One must be careful with definitions here, as the wrong sort of non-determinism can turn a very small class into NP. For example, depth-2 poly-size unbounded fan-in Boolean circuits can only recognize Π_2-$TIME(\log n)$. But if we give such a circuit both $\mathbf{x}$ and $\mathbf{y}$ inputs and say that it 'accepts' $\mathbf{x}$ iff there is some $\mathbf{y}$ such that the circuit accepts $\langle \mathbf{x}, \mathbf{y} \rangle$, it can recognise any language in NP.

We know the power of width-3 [Ba85] and width-5 PBP's – what of width-4? As S_4 is solvable, they cannot do all of NC^1 by the method used here for width-5, but we would like to prove they cannot do it at all. The conjecture of Section 7 would settle this, but 4-PBP's are a special case which might be more amenable to analysis.

We know that BP's without the permutation restriction require width-3 to do majority in poly-size [Ya83] and we know that width-5 suffices. Does the extra freedom to use instructions which aren't permutations help at all?

The fine structure of NC^1 is another good subject for further study. We know only that there are at least two classes (from [FSS81] and [Aj83]) but this is more than is known about most degree theories in complexity theory. Fagin et al. have made some progress [FKPS84], giving many AC^0 reducibilities among symmetric functions, but it appears that a new proof technique will be needed to settle the conjecture of Section 7 if it is true. AC^0-reducibility should also be compared with the projection reducibility of Valiant [SV81] in this setting. Majority is AC^0-complete for symmetric functions, but no function is projection-complete for them [GS??]. It is also interesting that an algebraically-defined language such as the word problem should be complete.

The unexpected power of NUDFA's suggests some foundational questions. Placing the power to recognize a language in a program to a very simple machine seems very different than placing it in, say, the state table of a Turing machine. How different is it, and how does it relate to other known models of computation?

9. Acknowledgements

This work is part of my Ph.D. research under the direction of Mike Sipser, who suggested this topic and advised me throughout my work on it. I would like to thank him for all his help and also thank Ravi Boppana, Johan Hastad, Philip Klein, Tom Leighton, and Leslie Valiant for various helpful discussions.

10. References

[Aj83] M. Ajtai, 'Σ_1^1 formulae on finite structures', Annals of Pure and Applied Logic 24 (1983), 1-48.

[ABHKST86] M. Ajtai, L. Babai, P. Hajnal, J. Komlós, E. Szemerédi, and G. Turán, 'Two lower bounds for branching programs', these proceedings.

[Ba85] D. A. Barrington, 'Width-3 permutation branching programs', Technical Memorandum TM-293, M.I.T. Laboratory for Computer Science.

[BCH84] P. W. Beame, S. A. Cook, and H. J. Hoover, 'Log-depth circuits for division and related problems', Proc. 25th IEEE FOCS, 1984, 1-6.

[BDFP83] A. Borodin, D. Dolev, F. E. Fich, and W. Paul, 'Bounds for width two branching programs', Proc. 15th ACM STOC, 1983, 87-93.

[Be73] C. H. Bennett, 'Logical reversibility of computation', IBM Journal of Research and Development, 17 (1973), 525-532.

[CFL83] A. K. Chandra, M. L. Furst, and R. J. Lipton, 'Multiparty protocols', Proc. 15th ACM STOC, 1983, 94-99.

[Co85] S. A. Cook, 'The taxonomy of problems with fast parallel algorithms', Information and Control 64 (Jan. 1985) 2-22.

[FKPS84] R. Fagin, M. M. Klawe, N. J. Pippenger, and L. Stockmeyer, 'Bounded depth, polynomial-size circuits for symmetric functions', IBM Report RJ 4040 (45198) (October 1983), IBM Research Laboratory, San Jose.

[FSS81] M. Furst, J. B. Saxe, and M. Sipser, 'Parity, circuits, and the polynomial time heirarchy', Proc, 22nd IEEE FOCS, 1981, 260-270.

[GS??] M. Geréb-Graus and E. Szemerédi, 'There are no p-complete families of symmetric Boolean functions', preprint.

[Ha86] J. Hastad, 'Improved lower bounds for small depth circuits', these proceedings.

[LF77] R. E. Ladner and M. J. Fischer, 'Parallel prefix computation', Proc. 1977 Intl. Conf. on Parallel Processing, 218-233.

[Le59] C. Y. Lee, 'Representation of switching functions by binary decision programs', Bell System Technical Journal 38 (1959) 985-999.

[Ma76] W. Masek, 'A fast algorithm for the string editing problem and decision graph complexity', M. Sc. Thesis, MIT, May 1976.

[Pu84] P. Púdlak, 'A lower bound on complexity of branching programs', Proc. Conference on the Mathematical Foundations of Computer Science, 1984, 480-489.

[Ru81] W. L. Ruzzo, 'On uniform circuit complexity', JCS 22, 3 (June 1981), 365-383.

[Sa72] J. Savage, 'Computation work and time on finit machines', J. ACM 19 (1972), 660-674.

[Sh85] J. B. Shearer, personal communication, 1985.

[SV81] S. Skyum and L. G. Valiant, 'A complexity theor based on Boolean algebra', Proc. 22nd IEEE FOCS, 1981, 244 253.

[Ya83] A. C. Yao, 'Lower bounds by probabilistic argu ments', Proc. 24th IEEE FOCS, 1983, 420-428.

[Ya85] A. C. Yao, 'Separating the polynomial-time heirar chy by oracles', 26th ACM STOC, 1985, 1-10.

[Za58] H. J. Zassenhaus, *The Theory of Groups*, 2nd ed (New York, Chelsea Publ. Co., 1958).