Width-Bounded Reducibility and Binary Search over Complexity Classes
(Extended Abstract)

Eric Allender*
Department of Computer Science
Rutgers University
New Brunswick, NJ 08903, USA

Christopher Wilson†
Department of Computer and Information Science
University of Oregon
Eugene, OR 97403, USA

SUMMARY

We introduce a notion of width-bounded reducibility. Width-bounded reducibility provides a circuit-based realization of Ruzzo-Simon-Tompa reducibility [RS-84], and allows us to generalize that notion of reducibility. We show that reductions of simultaneously restricted width and depth provide a characterization of binary search over complexity classes, as introduced by Wagner [Wa-89] and Buss and Hay [BH-88]. This allows us to present a circuit-based characterization of $P^{NP[\log]}$. Other results are presented that explore relationships among complexity classes, using width-bounded reductions as a tool.

*Supported in part by National Science Foundation Research Initiation Grant number CCR-8810467. Some of this research was performed while this author was a visiting professor at Institut für Informatik, Universität Würzburg, D-8700 Würzburg, Federal Republic of Germany.

†Supported in part by the National Science Foundation Grant number CCR-8810051

1 Introduction

Resource-bounded reducibility is one of the most important concepts of complexity theory. Time-bounded reductions [Co-71, LL-75] and space-bounded reductions [Jo-75, Si-77] were among the first to be studied. Then, in the context of circuit complexity, the notion of depth-bounded reducibility was seen to be of interest. Constant-depth reducibility was defined in [FS-84] and was studied in [CS-84]; NC_1 reducibility was defined in [Co-85]. More general NC_i and AC_i reductions were studied by Wilson [Wi-89]. (See also Chen [Ch-87].) NC Turing and many-one reducibility were contrasted in [Al-89].

In spite of the great deal of work done with depth-bounded reducibility, the analogous notion of width-bounded reducibility has not been considered before.

In this paper, we define a notion of width-bounded reducibility, and explore some of the properties of this reducibility. We show that width-bounded reducibility can be viewed as a suitable circuit-based realization of the space-

bounded reducibility introduced by Ruzzo, Simon, and Tompa in [RS-84]. Also, we show that reductions with simultaneous restrictions on depth and width yield a characterization of the notion of binary search over complexity classes that Wagner used in [Wa-89] to characterize subclasses of Δ_2^p. (An analogous notion was defined by Buss and Hay in [BH-88].) Thus, for example, we give a circuit-based characterization of the class $\mathrm{P}^{\mathrm{NP[log]}}$. We also answer some questions posed by Wagner by showing that $\mathrm{L}(2^{n^{O(1)}}) = \mathrm{P}$.

We believe that width-bounded reducibility may be useful in clarifying the relationships among various complexity classes. We present some first steps in this direction, by considering what consequences would follow if every set in L were reducible to a set in NC_1 via reductions of small width and depth.

In Section 2 we present our definition of width-bounded reducibility, and we present a "normal form" theorem for these reductions, which clarifies the relation between width-bounded reductions and Ruzzo-Simon-Tompa reducibility.

In Section 3 we review Wagner's notion of binary search over complexity classes, and characterize this sort of binary search in terms of reductions of bounded width and depth.

In section 4, we investigate width-bounded reducibility as a tool for proving relationships among complexity classes.

2 Width-Bounded Reducibility

A comparison of space with depth bounded reducibilities has been quite difficult [Wi-87, Wi-88]. The problem is that the depth bounded reducibilities seem too powerful. Their built-in advantage is that they have extra memory in the form of their wires. A query can depend on the outcomes of many earlier queries simply by routing the wires appropriately. In order to directly simulate such a reduction, a space bounded Turing machine is forced to write down the outcomes of earlier queries, thus violating its space bound.

There are two natural approaches to this problem. One can increase the power of a space bounded reducibility, as is done in [Wi-88] and [Bu-88]. Alternatively, one could restrict the inherent memory of circuits by bounding their width, as we do here.

The notion of circuit width must be defined carefully, in order to relate width to complexity classes. Pippenger [Pi-79] defined the appropriate notion of circuit width, and proved that SC (=DTIME,SPACE($n^{O(1)}, \log^{O(1)} n$)) is equal to the class of sets that can be recognized by uniform families of circuits of polynomial size and polylogarithmic width. It should be noted that in Pippenger's definition of width, each bit of the input is made available as often as it is requested, without cost. If each bit were made available only once, then even some sets with very low complexity would require large width.

The notion of width-bounded reducibility that we consider is analogous to the depth-bounded reducibility studied in [FS-84, CS-84, Co-85, Wi-89]. A reduction is a uniform family of circuits with *oracle gates*. Just as the depth of an oracle gate must be defined precisely in order to define NC_i or AC_i reductions, so must we be careful to define the width of an oracle gate.

Since many of our results deal with very small complexity classes such as AC_0, it is important that we be careful in the notion of

uniformity that we use. Following the lead of Buss [Bu-87] and Barrington, Immerman, and Straubing [BI-88], we will use logtime uniformity throughout this paper. Definitions of logtime uniformity may be found in [Bu-87] and [BI-88].

Definition: A *reduction* is a uniform family of circuits $\{C_n : n \in \mathbf{N}\}$ where each circuit C_n has n inputs, and consists of AND gates, OR gates, NOT gates, and oracle gates. (In addition to the n inputs, the negations of these inputs, as well as the constants 0 and 1 are also provided.) An *oracle gate* for a language L is a gate which takes a bit string as input and outputs 1 iff the bit string is a string in L. We say that A is *reducible to L via* $\{C_n\}$ if the only oracle gates in $\{C_n\}$ are oracle gates for L, and each circuit C_n accepts $A^{=n}$.

The following definitions are standard. (See, e.g., [DC-89].)

Definition: The *size* of a circuit is the number of gates in the circuit. The *depth* of a gate in a circuit is the length of the longest path from an input to the gate; the depth of a circuit is the maximum of the depth of the gates in the circuit. The *width* of a circuit is $\max\{m : \exists d$ there are m gates with depth at most d that are input to gates at level greater than $d\}$.

Note in particular, that only wires from *gates* are counted when computing the width of a circuit; connections to inputs (or to the negations of inputs or to the constants 0 and 1) are "free". This is also true for oracle gates. That is, the width of an oracle gate is the number of inputs to the oracle gate that *are not inputs to C_n*. This corresponds to a restriction on the "work-space" of the circuit. Wires from gates into other gates form a sort of memory, and this is not accurately measured by the standard size and depth bounds.

In certain places, we will have need to refer to other notions of reducibility. *Logtime reductions* are very closely related to logtime uniformity, and are defined in [Bu-87]. AC_0 reductions are (logtime uniform) reductions of polynomial size and $O(1)$ depth. For any set A, L^A will denote the class of languages reducible to A via logspace-computable Turing reductions, as considered by e.g. [RS-84,Wa-89]; in this notion of logspace-bounded Turing reducibility, the oracle tape is not included in the space-bound (and thus queries of polynomial length can be asked). See also [LL-76, Ha-88] for a discussion of logspace-bounded Turing reducibility.

We will find it useful to prove a "normal-form" theorem for width-bounded reductions. Let us say that a reduction $\{C_n\}$ is *in normal form* if, for each circuit C_n, the input to each oracle gate in C_n is of the form xw, where x is the input to C_n. The *width* of such a gate is the length of w.

Theorem 1 Let $\mathcal{C}$ be a complexity class closed under logtime reductions. If L is reducible to a set in $\mathcal{C}$ via a reduction with size $s(n)$, depth $d(n)$ and width $w(n)$, then L is reducible to a set in $\mathcal{C}$ via a reduction in normal form of the same size and depth, and width $w(n) + O(\log s(n))$.

Proof: Let L be reducible to A via $\{C_n\}$, where A is a set in $\mathcal{C}$. Define the function f as follows: $f(x, g, b_1 \ldots b_r)$ is the word that is input to gate g of $C_{|x|}$, if the outputs of the gates which are input to g are given by $b_1 \ldots b_r$. By the uniformity of $\{C_n\}$, f is a logtime reduction. Let A' be the set $\{x\#g\#b_1 \ldots b_r : f(x, g, b_1 \ldots b_r)$ is in $A\}$. Then A' is in $\mathcal{C}$, and L is reducible to A' via a reduction of exactly the same size and depth as $\{C_n\}$. The width of the new reduction is increased by the size of the string g. ∎

Throughout the rest of this paper, we will assume that all reductions are in normal form.

Comparing this "normal form" theorem with Lemma 7 of [RS-84], we can see that Ruzzo-Simon-Tompa reducibility corresponds exactly to reductions with oracle gates of logarithmic width.

In analogy with [Wi-89], where NC_i and AC_i reductions are studied, it is possible to use width-bounded reductions to define SC_i reducibility. Recall that Pippenger showed in [Pi-79] that SC is equal to the class of languages accepted by uniform circuits of polynomial size and width $\log^{O(1)} n$. Define SC_i to be the class of languages accepted by uniform circuits of polynomial size of width $O(\log^i n)$. Similarly, define SC_i reductions to be reductions of polynomial size and width $O(\log^i n)$. For any class of sets $\mathcal{C}$, let $\mathrm{SC}_i^{\mathcal{C}}$ denote the set of languages reducible to sets in $\mathcal{C}$ via SC_i reductions.

Proposition 2 For all i and j, $\mathrm{SC}_i^{\mathrm{SC}_j} = \mathrm{SC}_{\max(i,j)}$.

Proof: It is evident that $\mathrm{SC}_{\max(i,j)} \subseteq \mathrm{SC}_i^{\mathrm{SC}_j}$. We shall illustrate how to show that $\mathrm{SC}_i^{\mathrm{SC}_j} \subseteq \mathrm{SC}_{\max(i,j)}$. Suppose $A \in \mathrm{SC}_i^{\mathrm{SC}_j}$, and we have a circuit α recognizing A on inputs x of length n; α will have width $c \log^i n$ and ask normal form queries to an oracle $B \in \mathrm{SC}_j$. In building the circuit α' for A without queries, any query xw will be replaced by a circuit β_w of polynomial size and width at most $d \log^j n$. Consider some level in α making queries $xw_1, \ldots, xw_k$, $\sum_{l=1}^{k} |w_l| \leq c \log^i n$. Query xw_1 will be replaced by β_{w_1}, while the replacement of other queries will be deferred till deeper in the circuit. After giving β_{w_1} sufficient depth to finish, we replace query xw_2 with β_{w_2}, deferring the rest, and so on. The width of this subsection is at most $k + c \log^i n + d \log^j n = O((\log n)^{\max(i,j)})$, which also bounds the width of α'. The size of α' is still polynomial and the circuit family thus derived can be constructed uniformly, so $A \in \mathrm{SC}_{\max(i,j)}$. ∎

Pippenger showed that deterministic logspace can be simulated by (uniform) circuits of logarithmic width [Pi-79]. However, it is not known if logspace is sufficient in turn to simulate circuits of this sort. The problem is that it may be difficult for a logspace-bounded machine to compute a width-efficient layout of a circuit. On the other hand, if we modify our uniformity condition to require that the *depth* of a gate be part of its name, then such a simulation is possible. This motivates the following definition:

Definition: A uniform circuit family is *layout-uniform* if (1) the depth of each gate is part of the name of each gate, and (2) no gate at any depth d is connected to any gate at depth less than $d - 1$.

Proposition 3 L^A is the class of languages reducible to A via layout-uniform reductions of polynomial size and logarithmic width.

3 Binary Search

Wagner [Wa-89] studies the classes $\mathrm{NP}(b(n))$ which correspond to a sort of binary search over witnesses for strings in sets in NP; the bound $b(n)$ is the range over which the search is allowed to take place. (A very similar notion was defined in Section 4.1 of [BH-88].) Wagner's interest, as well as that of Buss and Hay, was in the classification of optimization problems in NP using Krentel's framework [Kr-88], and they were also interested in characterizing classes defined by restricted queries

to NP, such as the Boolean hierarchy [CG-88], $P^{NP[\log]}$, and P^{NP}.

Definition: For a class of languages $\mathcal{C}$ and integer function r, define $\mathcal{C}(r(n))$ to be the set of languages A such that, for some $L \in \mathcal{C}$, $x \in A \iff (\max\{i \leq r(|x|) : \langle x, i\rangle \in L\}$ is odd), where in addition, L satisfies the property that $\langle x, i\rangle \in L \Longrightarrow \langle x, i-1\rangle \in L$. (This last condition on L is not necessary when $\mathcal{C}$ = NP, but is necessary when $\mathcal{C}$ is some other complexity class, such as L, P, etc.)

The intuitive idea behind the definition of $\mathcal{C}$ is that each string x is associated with a set of "witnesses," and we are interested in the complexity of finding the largest witness. (Note that this is equivalent to doing binary search over a given range.) It is shown in [Wa-89, BH-88] that various levels of the Boolean Hierarchy can be characterized in terms of $\mathrm{NP}(k)$ for constant k; it is also shown that $P^{NP[\log]} = \mathrm{NP}(n^{O(1)})$, and $\mathrm{NP}(2^{n^{O(1)}}) = P^{NP}$.

Wagner [Wa-89] noted that the same notion could be defined for classes other than NP; he observed that $\mathrm{L}(n^{O(1)}) = \mathrm{L}$, $\mathrm{NL}(n^{O(1)}) = \mathrm{NL}$, and $\mathrm{P}(n^{O(1)}) = \mathrm{P}(2^{n^{O(1)}}) = \mathrm{P}$, and he asked what could be said about $\mathrm{L}(2^{n^{O(1)}})$.

We answer this question below, by considering reductions of restricted depth and width.

Note that a reduction of restricted depth and width necessarily has small size; similarly, a reduction of very small size must have small depth and width. Our characterization of $\mathcal{C}(r(n))$ finds its best expression when given in terms of size, rather than in terms of depth and width; however, our normal form theorem for width-bounded reductions is quite useful.

Theorem 4 For any complexity class $\mathcal{C}$ closed under AC_0 reductions, $\mathcal{C}(O(b(n)))$ is equal to the class of languages which are reducible to languages in $\mathcal{C}$ via reductions of size $\log b(n) + O(1)$.

Proof: ($\subseteq$) This direction is quite easy. Let L be the set in $\mathcal{C}$ such that $x \in A \iff \max(i \leq b(n) : \langle x, i\rangle \in L)$ is odd. First query if $\langle x, 10\ldots0\rangle$ is in L; call the answer to this query b_1. Next query if $\langle x, b_1 1\ldots0\rangle \in L$. It is clear how to proceed. Note that exactly $\lceil \log b(n)\rceil$ gates are necessary. (The width of the reduction is also exactly $\lceil \log b(n)\rceil$.)

($\supseteq$) Let A be reducible to a set B in $\mathcal{C}$ via a reduction $\{C_n\}$ of size $s(n)$. Assume without loss of generality that the representation of each circuit C_n is topologically sorted. Let L be the set of strings $\langle x, z\rangle$ such that x has length n and z is lexicographically before y, where y is a binary string of length $s(n)$ which records, for each gate, the value it takes on when x is given as input to C_n. We show below that L is AC_0 reducible to B, and thus L is in $\mathcal{C}$. Note that x is in A iff the last bit of y is 1. Thus A is in $\mathcal{C}(2^{s(n)})$.

It remains only to show that L is AC_0 reducible to B. First note that for each i there is a small AC_0 circuit (possibly containing oracle gates for B) that checks if the i-th gate takes on value $z[i]$ when the inputs to the i-th gate take on the values given by the corresponding $z[j]$'s, for $j < i$. Let us say that $z[i]$ is *consistent* if this is the case. Now it suffices to note that $\langle x, z\rangle \in L$ iff ($\forall j$ $z[j]$ is consistent) or ($\exists i$ $z[i]$ is not consistent and $z[i] = 0$ and $\forall j < i$, $z[j]$ is consistent). ∎

That is, in some sense, binary search is characterized by simultaneous restrictions on depth and width. This theorem allows us to answer the question posed by Wagner, concerning $\mathrm{L}(2^{n^{O(1)}})$.

Corollary 5 $AC_0(2^{n^{O(1)}}) = \mathrm{L}(2^{n^{O(1)}}) = \mathrm{P}$.

The requirement that $\mathcal{C}$ be closed under AC_0 reductions is reasonable for most interesting subclasses of P, but it seems not to apply to other interesting classes, such as NP. However, using special properties of NP, the same result can be shown to hold.

Proposition 6 Let $b(n)$ be bounded by a polynomial in n. Then $NP(O(b(n)))$ is equal to the class of languages which are reducible to languages in NP via reductions of size $\log b(n) + O(1)$.

(For superpolynomial functions $b(n)$, this claim is not known to hold.)

Proof: ($\subseteq$) This direction is proved as in the preceding theorem. (Note that closure under AC_0 reductions was not used there.)

($\supseteq$) Let L be reducible to a set in NP via a reduction of size $\log b(n) + O(1)$. Note that the gates in the reduction can take on at most $O(b(n))$ different values, and thus at most $O(b(n))$ queries of the form $\langle xw \rangle$ are possible on inputs x of length n. Thus L is reducible to SAT via a polynomial-time truth-table reduction asking at most $O(b(n))$ queries, and by [Wa-89], $L \in NP(O(b(n)))$. ∎

Thus we obtain circuit characterizations of $P^{NP[\log]}$.

Corollary 7 L is in $P^{NP[\log]}$ iff L is reducible to a set in NP via a reduction of size $O(\log n)$.

We find Corollary 7 interesting, since, for example, it is not known what the class of languages NC_1 or AC_1 reducible to NP is. Wagner [Wa-89] shows that $P^{NP[\log]} = L^{NP}$, but using his notion of logspace-Turing reductions, it is not always the case that $NC_1^A \subseteq L^A$; (see, for example, [Wi-87, Wi-88]). Thus this does not imply that $NC_1^{NP} = P^{NP[\log]}$. Indeed, it is not too hard to show that $L^{NP} \subseteq AC_0^{NP}$. However, Corollary 7 does give a circuit-based characterization of $P^{NP[\log]}$; we believe that it is the first such characterization.

(In light of Proposition 3, the result that $P^{NP[\log]} = L^{NP}$ gives another characterization of $P^{NP[\log]}$; namely $P^{NP[\log]}$ is the class of sets reducible to SAT via reductions of polynomial size and logarithmic width. That is, with NP oracles, logarithmic size is as good as polynomial size and logarithmic width.)

4 Comparing Complexity Classes

Reductions with simultaneous restrictions on width and depth are very weak. Note, for example, that PARITY is both NC_1 and SC_1 reducible to AC_0, but we show below that PARITY cannot be reduced to any set in AC_0 via any reduction with simultaneous bounds on depth and width less than inverse polynomial. Because of the very weakness of these reductions, we feel that they may prove useful in demonstrating new relationships among certain closely-related complexity classes.

Proposition 8 Every language in $AC_0(s(n))$ is accepted in depth $O(1)$ and size $s(n)n^{O(1)}$.

Corollary 9 If PARITY is in $AC_0(b(n))$, then $b(n) = \Omega(2^{n^\epsilon})$ for some $\epsilon > 0$.

Because reductions of small size and depth are so weak, it would be interesting if they could be used to show new relationships among important complexity classes. For example, the question of the relationship between NC_1 and L is of considerable interest.

We were initially intrigued by the possibility that L could be reducible to NC_1 via reductions of depth and width $\log^{O(1)} n$. The following observation initially seemed to indicate to us that such a result would be unlikely.

Observation: If L is contained in $NC_1(2^{\log^{O(1)} n})$, then every branching program of polynomial size is equivalent to a branching program of width 6 and length $2^{\log^{O(1)} n}$.

On the other hand, it follows from [Ba-89] that every set with circuits of depth $O(\log^2 n)$ (including not only L, but also NL and NC_2) is recognized by a branching program of width 5 and length $2^{O(\log^2 n)}$; thus perhaps there is still some hope of showing L is contained in $NC_1(2^{\log^{O(1)} n})$.

Acknowledgments
The first author acknowledges discussions with David Mix Barrington; thanks are also due to Pierre McKenzie and Denis Thérien for organizing the 1990 Barbados Workshop on Complexity Theory, where the discussions took place.

References

[Al-89] E. Allender, *P-uniform circuit complexity*, J. ACM 36, 912–928.

[Ba-89] D. A. Barrington, *Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1*, J. Computer and System Sci. 38, 150–164.

[BI-88] D. A. Mix Barrington, N. Immerman, and H. Straubing, *On uniformity within NC^1*, to apear in Information and Computation. A preliminary version appeared in Proc. 3rd IEEE Structure in Complexity Theory Conference, pp. 47–59.

[Bu-87] S. Buss, *The Boolean formula value problem is in ALOGTIME*, Proc. 19th Annual ACM Symposium on Theory of Computing, pp. 123–131.

[Bu-88] J. Buss, *Relativized alternation and space-bounded computation*, JCSS 36, 351–378.

[BH-88] S. Buss and L. Hay, *On truth-table reducibility and the difference hierarchy over NP*, Proc. 3rd IEEE Conference on Structure in Complexity Theory, pp. 224–233.

[CG-88] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung, *The Boolean Hierarchy I: Structural Properties*, SIAM J. Comput 17, 1232–1253.

[CS-84] A. K. Chandra, L. J. Stockmeyer, U. Vishkin, *Constant depth reducibility*, SIAM J. Comput. 13, 423-439.

[Ch-87] J. Chen, *Logarithmic depth reducibility and the NC hierarchy*, manuscript.

[Co-71] S. Cook, *The complexity of theorem proving procedures*, Proc. 3rd Annual ACM Symposium on Theory of Computing, pp. 151–158.

[Co-85] S. Cook, A taxonomy of problems with fast parallel algorithms, *Information and Control* 64 (1985) 2–22.

[DC-89] P. Dymond and S. Cook, *Complexity theory of parallel time and hardware*, Information and Computation 80, 205–226.

[FS-84] M. Furst, J. Saxe, M. Sipser, *Parity, circuits, and the polynomial-time hierarchy*, Mathematical Systems Theory 17, 13–27.

[Hå-86] J. Håstad, *Almost optimal lower bounds for small depth circuits*, Proc. 18th ACM Symposium on Theory of Computing, pp. 6–20.

[Ha-88] J. Hartmanis, *New developments in structural complexity theory*, Proc. 15th International Colloquium on Automata, Languages, and Programming, Lecture Notes in Computer Science 317 (1988), pp. 271–286.

[Jo-75] N. D. Jones, *Space-bounded reducibility among combinatorial problems*, J. Comput. and System Sci. 11, 68-85.

[Kr-88] M. Krentel, *The complexity of optimization problems*, J. Comput. and System Sci. 36, 490–509.

[LL-75] R. Ladner, N. Lynch, and A. Selman, *A comparison of polynomial-time reducibilities*, Theoretical Computer Science 1, 103–123.

[LL-76] R. Ladner and N. Lynch, *Relativization of questions about log space computability*, Mathematical Systems Theory 10, 19–32.

[Pi-79] N. Pippenger, *On simultaneous resource bounds*, Proc. 20th IEEE Symposium on Foundations of Computer Science, pp. 307-311.

[RS-84] Walter Ruzzo, Janos Simon, and M. Tompa, *Space-bounded hierarchies and probabilistic computation*, J. Comput. and System Sci. 28, 216–230.

[Si-77] Istvan Simon, *On some subrecursive reducibilities*, PhD Dissertation, Stanford University, 1977.

[Wa-89] K. Wagner, *Bounded query classes*, to appear in SIAM J. Comput. A preliminary version appeared as *On restricting access to an NP-oracle*, Proc. 15th International Colloquium on Automata, Languages, and Programming, Lecture Notes in Computer Science 317 (1988), pp. 682–696.

[Wi-87] C. B. Wilson, *Relativized NC*, Math. Systems Theory 20, 13–29.

[Wi-88] C. B. Wilson, *A measure of relativized space which is faithful with respect to depth*, J. Comput. and System Sciences 36, 303–312.

[Wi-89] C. B. Wilson, *Decomposing NC and AC*, to appear in SIAM J. Comput. A preliminary version appeared in Proc. 4th IEEE Conference on Structure in Complexity Theory, pp. 124–131.