A Note on Nondeterminism in Small, Fast Parallel Computers

IAN PARBERRY

Abstract—**Nondeterministic analogues of the well-known classes NC and SC, called NNC and NSC, respectively, are investigated. Adding nondeterminism to SC leaves it in the domain of parallel computation since NSC $\subseteq$ POLYLOGSPACE. That is, NSC is a subset of the class of languages computable by fast parallel computers. Adding nondeterminism to NC appears to make it much more powerful since NNC = NP. It is clear that NSC $\subseteq$ NNC, and probable that NSC $\subset$ NNC. We provide further evidence for this conjecture by showing that NSC is precisely the class of languages recognizable in simultaneous polynomial time and polylog reversals by a nondeterministic Turing machine with a read-only input tape and a single read-write work-tape. It is known that NNC is precisely the class of languages recognizable in simultaneous polynomial time and polylog reversals by a nondeterministic Turing machine with a read-only input tape and two read-write work-tapes.**

Index Terms—**Crossing sequence, NC, nondeterminism, parallel computation thesis, parallel computers, reversal, SC, simultaneous resource bounds, space, time.**

I. Introduction

The complexity classes NC and SC have been well studied in the recent literature. NC is the class of languages which can be accepted by small, fast parallel computers. NC remains the same class under many definitions of a "parallel computer" (see, for example, Dymond [4], Dymond and Cook [5], and Ruzzo [20]). Pippenger [19] has demonstrated that NC is the class of languages which can be recognized by a deterministic Turing machine in polynomial time and polylog reversals (a *reversal* is said to occur when any tape head changes direction). SC is the class of languages which can be recognized by a deterministic Turing machine in polynomial time and polylog work-space (that is, not counting the space needed for the storage of the inputs). It is widely conjectured that SC $\neq$ NC. More strongly, it is expected that NC $\not\subseteq$ SC and SC $\not\subseteq$ NC (see, for example, Cook [3]).

We will consider the impact of nondeterminism on small, fast parallel computers. The effect of nondeterminism on sequential computers is well studied in the form of the P $\neq$ NP problem [2], [13]. P is the class of languages recognized by fast sequential computers. P remains the same class under many definitions of "computer." NP is the nondeterministic analogue of P. While the P $\neq$ NP question is still open, a large amount of evidence in favor of the conjecture has been accumulated. There exist a large number of problems which have been shown to be *NP-complete;* that is, they are members of NP and have the property that if any one of them are members of P, then P = NP. Garey and Johnson [6] have gathered a large list of NP-complete problems.

Nondeterministic analogues of the classes NC and SC, called NNC and NSC, respectively, can be defined in a natural manner using a standard nondeterministic Turing machine model with k one-way infinite work-tapes and a single read-only input tape (after the manner of Aho, Hopcroft, and Ullman [1]). A nondeterministic Turing machine runs in time $T(n)$ and space $S(n)$ if for all accepted inputs of size n there is an accepting computation which uses time at most $T(n)$ and space at most $S(n)$. A nondeterministic Turing machine runs in time $T(n)$ and reversals $R(n)$ if for all accepted inputs of size n there is an accepting computation which uses time at most $T(n)$ and reversals at most $R(n)$. The reversals and space are measured on the work-tapes only. We assume that the running time $T(n) = \Omega(n)$ and the space bound $S(n) = \Omega(\log n)$. For definiteness, we define NC to be the class of languages recognizable in polynomial time and polylog reversals on a k-tape deterministic Turing machine, NNC to be the class of languages recognizable in polynomial time and polylog reversals on a k-tape nondeterministic Turing machine, SC to be the class of languages recognizable in polynomial time and polylog space on a k-tape deterministic Turing machine, NSC to be the class of languages recognizable in polynomial time and polylog space on a k-tape nondeterministic Turing machine.

The proof of Pippenger that NC is exactly the class of languages recognizable in polynomial size and polylog depth by uniform circuits [19] extends easily to our Turing machine model. The simulation of a uniform circuit by a Turing machine is straightforward since our Turing machine is a more general form of that used by Pippenger in that we do not count reversals on the input tape. Pippenger does not count the number of tapes required; two work-tapes are sufficient. Dymond [4] notes that three tapes are sufficient, but he does not distinguish between the input tape and the work-tapes. The simulation of our Turing machine by a uniform circuit appears in Parberry [15], [16]. An improved simulation with tighter resource bounds appears in Parberry [17], [18]. Suppose we define a nondeterministic uniform circuit to be a uniform circuit augmented with a polynomial number of auxiliary inputs. Such a circuit accepts its input if there is an assignment to the auxiliary inputs that results in the output having value 1. It is easy to show that NNC is exactly the class of languages recognizable in polynomial size, polylog depth, by a nondeterministic uniform circuit family.

Let POLYLOGSPACE denote the class of languages recognizable in polylog space on a Turing machine. By Savitch's theorem [21], the Turing machine may be either deterministic or nondeterministic. POLYLOGSPACE is widely accepted to be the class of languages recognizable by fast parallel computers. Goldschlager [7], [9] has called this characterization the *parallel computation thesis.* NC is widely accepted to be the class of languages recognizable by small, fast parallel computers. Dymond [4], [5] has called this characterization the *extended parallel computation thesis.* By Savitch's theorem [21], NSC $\subseteq$ POLYLOGSPACE; that is, languages in NSC can be recognized by fast parallel computers.

II. The Results

Although it is popularly conjectured that SC $\not\subseteq$ NC, the opposite is true for the corresponding nondeterministic complexity classes. To see that NSC $\subseteq$ NNC, note that NSC $\subseteq$ NP and:

Theorem 1: NNC = NP.

Proof: Clearly NNC $\subseteq$ NP. To see that NP $\subseteq$ NNC, first observe that every language in P has a polynomial-size uniform circuit. This fact was first noted by Ladner [14]. (A more detailed proof appears in Parberry [18].) We will first demonstrate that P $\subseteq$ NNC. Suppose $L \in$ P. From the above observation, there exists a polynomial-size uniform circuit C_L which recognizes L. A polynomial-size, logarithmic-depth nondeterministic uniform circuit for L can be constructed as follows. The nondeterministic circuit guesses a value for every wire in C_L and verifies that the guessed values are consistent and that the output is 1. The verification can be carried out by using a single layer of gates which corresponds exactly to the gates in C_L, and a tree of AND gates of depth $O(\log n)$. This shows that P $\subseteq$ NNC. A similar argument shows that NP $\subseteq$ NNC. If $L \in$ NP, then L can be recognized in polynomial time by a deterministic Turing machine augmented with a "random" tape provided we define acceptance of the input to mean that there exists a sequence of symbols which, when written on the random tape, causes the machine to deterministically enter the accept state on the given input. This is essentially the "guess-verify" Turing machine model of Garey and Johnson [6]. A polynomial-size nondeterministic uniform circuit can be constructed from this Turing machine by applying the technique of Ladner to the deterministic portion of the computation. This can be simulated by a polynomial-size, logarithmic-depth nondeterministic uniform circuit as before. □

In light of Theorem 1, it appears extremely likely that adding nondeterminism to small, fast parallel computers increases their power. Since NC $\subseteq$ P, if NNC = NC then NP = P = NC. It is

Manuscript received September 10, 1986; revised October 29, 1987.

The author is with the Department of Computer Science, The Pennsylvania State University, University Park, PA 16802.

IEEE Log Number 8825691.

widely conjectured that P $\neq$ NP (that is, there are problems which are feasible to check on a sequential computer, but not feasible to solve), and that P $\neq$ NC (that is, there are problems which can be feasibly solved on a sequential computer, but which cannot be solved on a small, fast parallel computer). As mentioned above, evidence is provided for the former conjecture by a large number of NP-complete problems. In a similar manner, evidence is provided for the latter conjecture by a large number of P-complete problems (see, for example, [8], [10], [11], and [14]). If any P-complete problem can be demonstrated to be a member of NC, then P $\subseteq$ NC.

NSC and NNC superficially appear different since NSC remains the same even if only a single work-tape is allowed (by Theorem 7.2 of [12]) while in contrast it appears that two work-tapes are necessary to capture all of NNC. Certainly two tapes are sufficient, by use of the "reduction to sorting" technique of Pippenger [19]. We will show that NSC is precisely the class of languages recognizable in simultaneous polynomial time and polylog reversals by a nondeterministic Turing machine with a read-only input tape and a single read–write work-tape. We will call the latter class NNC_1.

Theorem 2: A nondeterministic single work-tape Turing machine which runs in time $T(n)$ and space $S(n)$ can be simulated by a nondeterministic single work-tape Turing machine in time $O(T(n)\cdot S(n)^2$ and reversals $O(S(n))$.

Proof: (Sketch) The simulation takes place in four phases.

1) In a single sweep, nondeterministically guess a sequence of configurations and a sequence of rules, interleaved so that each rule precedes the configuration to which it is to be applied. In the process, nondeterministically mark each cell in the configuration as either occupied or unoccupied by the tape head.
2) Verify in a single sweep that the work-tape head is marked as occupying exactly one tape cell in each configuration, and that the final configuration has the accept state.
3) Verify in a constant number of sweeps that the sequence of rules is consistent with the input, the guessed sequence of states, and the symbols under the head in each configuration.
4) Verify that the head movements are consistent with the guessed rules. Initially all cells are unmarked. On the ith sweep, $1 \leq i \leq S(n)$, do the following. When entering a configuration:
   a) Remember what rule is to be applied to get to the next configuration.
   b) Find the ith cell in the configuration (it will be the first unmarked cell). Remember whether the head was in the $(i-1)$st cell when passing over it.
   c) If the head was on the ith cell of the previous configuration, then do the following, else do nothing. If the head movement implied by the previous rule was "move left" and the $(i-1)$th cell was not scanned by the tape head, then fail. Otherwise, if there was no head movement, verify that the ith cell is scanned by the head. Otherwise, if the head movement was "move right," remember to check the $(i+1)$th tape cell when leaving the configuration in step d) below. Check that the symbol in cell i is consistent with the current rule.
   d) Mark the ith cell as visited, and move to the next configuration.

The running time is dominated by Phase 4 which requires $S(n)$ sweeps, each taking time $O(T(n)\cdot S(n))$ and constant reversals. □

Corollary 3: NSC $\subseteq$ NNC_1.

Theorem 4: A nondeterministic single work-tape Turing machine which runs in time $T(n)$ and reversals $R(n)$ can be simulated by a nondeterministic Turing machine in time $O(T(n)\cdot\log T(n))$ and space $O(R(n)\cdot\log T(n))$. □

Proof. (Outline) The Turing machine guesses a sequence of crossing sequences, verifying after each guess that the new crossing sequence matches its predecessor. The latter can be tested using an algorithm similar to that in Section 2.6 of Hopcroft and Ullman [12]. Our crossing sequences must contain, along with the sequence of states in which cell boundaries were crossed, a time stamp and a positive integer which indicates in the position of the input head. Thus, each crossing sequence must have length $O(R(n)\cdot\log T(n))$, which gives a space bound of $O(R(n)\cdot\log T(n))$. The running time is linear in the total length of the series of crossing sequences, that is, $O(T(n)\cdot\log T(n))$. □

Corollary 5: NNC_1 $\subseteq$ NSC.

III. Conclusion

While it is clear that NSC $\subseteq$ NNC, it is unlikely that the two classes are equal. We have provided further evidence that this is the case by showing that NSC is precisely NNC restricted to one work-tape; two work-tapes are sufficient and appear necessary to capture all of NNC. Nondeterministic analogues of standard parallel complexity classes are intrinsically interesting. For example, NSC is a subset of the class of languages recognized by fast parallel computers, while NNC is exactly NP. It appears that adding nondeterminism to small, fast parallel computers increases their power.

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms.* Reading, MA: Addison-Wesley, 1974.

[2] S. A. Cook, "The complexity of theorem proving procedures," in *Proc. 3rd ACM Symp. Theory Comput.,* 1971, pp. 151–158.

[3] ——, "Towards a complexity theory of synchronous parallel computation," *L'Enseignement Mathematique,* vol. 30, 1980.

[4] P. W. Dymond, "Simultaneous resource bounds and parallel computations," Ph.D. dissertation, Tech. Rep. TR145/80, Dep. Comput. Sci., Univ. of Toronto, Aug. 1980.

[5] P. W. Dymond and S. A. Cook, "Hardware complexity and parallel computation," in *Proc. 21st Annu. IEEE Symp. Foundations Comput. Sci.,* Oct. 1980, pp. 360–372.

[6] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* San Francisco, CA: Freeman, 1979.

[7] L. M. Goldschlager, "Synchronous parallel computation," Ph.D. dissertation, Tech. Rep. TR-114, Dep. Comput. Science, Univ. of Toronto, Dec. 1977.

[8] ——, "ϵ-productions in context-free grammars," *Acta Informatica,* vol. 16, no. 3, pp. 303–308, 1981.

[9] ——, "A universal interconnection pattern for parallel computers," *J. ACM,* vol. 29, pp. 1073–1086, Oct. 1982.

[10] L. M. Goldschlager and I. Parberry, "On the construction of parallel computers from various bases of Boolean functions," *Theor. Comput. Sci.,* vol. 43, pp. 43–48, May 1986.

[11] L. M. Goldschlager, R. A. Shaw, and J. Staples, "The maximum flow problem is log space complete for P," *Theor. Comput. Sci.,* vol. 21, no. 1, pp. 105–111, 1982.

[12] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages and Computation.* Reading, MA: Addison-Wesley, 1979.

[13] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations,* J. W. Thatcher, Ed. New York: Plenum, 1972.

[14] R. E. Ladner, "The circuit value problem is log space complete for P," *SIGACT News,* vol. 7, no. 1, pp. 18–20, 1975.

[15] I. Parberry, "A complexity theory of parallel computation," Ph.D. dissertation, Dep. Comput. Sci., Univ. of Warwick, May 1984.

[16] ——, "Some practical simulations of impractical parallel computers," in *VLSI: Algorithms and Architectures,* P. Bertollazzi and F. Lucio, Eds. in *Proc. Int. Workshop Parallel Comput. VLSI,* North-Holland, 1985, pp. 27–37.

[17] ——, "An improved simulation of space and reversal bounded deterministic Turing machines by width and depth bounded uniform circuits," *Inform. Proc. Lett.,* vol. 24, pp. 363–367, Apr. 1987.

[18] ——, *Parallel Complexity Theory, Research Notes in Theoretical Computer Science.* London, England: Pitman, 1987.

[19] N. Pippenger, "On simultaneous resource bounds," in *Proc. 20th Annu. IEEE Symp. Foundations Comput. Sci.,* Oct. 1979, pp. 307–311.

[20] W. L. Ruzzo, "On uniform circuit complexity," *J. Comput. Syst. Sci.,* vol. 22, pp. 365–383, June 1981.

[21] W. J. Savitch, "Relationships between nondeterministic and deterministic tape complexities," *J. Comput. Syst. Sci.,* vol. 4, no. 2, pp. 177–192, 1970.