We present a deterministic algorithm running in space $O(\log^2 n / \log \log n)$ solving the connectivity problem on strongly unambiguous graphs. In addition, we present an $O(\log n)$ time-bounded algorithm for this problem running on a parallel pointer machine.

1 Introduction

One of the most central questions of complexity theory is to relate determinism and nondeterminism. Our inability to exhibit the precise relationship between these two notions motivates the investigation of intermediate notions such as symmetry or unambiguity. In this paper we concentrate on unambiguity in space-bounded computation, and present improved deterministic and parallel simulations.

Recently, surprising results have indicated that "symmetric" space bounded computation is weaker than nondeterminism. In particular, symmetric logspace has been shown to be contained in parity logspace [12], in $SC^2$ [18], and in $DSPACE(\log^{1.5} n)$ [19]. None of these upper bounds is known to hold in the nondeterministic case. If we consider these questions for space bounded unambiguous classes, we are confronted with the fact that there are several ways to define notions of unambiguity that apparently do not coincide [3]. In this paper we will concentrate on the notion of unambiguity (in the sense of unique existence of computation paths), and of strong unambiguity (in the sense of

* Supported in part by NSF grant CCR-9509603.
** Supported in part by NSF grant CCR-9509603.
uniqueness of computations between any pair of configurations). This yields the two classes \( \text{USPACE}(\log n) \) and \( \text{StUSPACE}(\log n) \).

By definition, \( \text{USPACE}(\log n) \) is a subclass of parity logspace; this is not known to hold for \( \text{NSPACE}(\log n) \) (although see [23]); there is no additional nontrivial containment known for \( \text{USPACE}(\log n) \). However, \( \text{StUSPACE}(\log n) \) is contained in \( SC^2 \), since strongly unambiguous logspace languages can be accepted by deterministic auxiliary pushdown automata in polynomial time [3, 5]. Still, it was unknown whether there are \( o(\log^2 n) \) space algorithms for strongly unambiguous logspace languages. We answer this question affirmatively by showing that \( \text{StUSPACE}(\log n) \) is contained in \( \text{DSPACE}(\log^2 n/ \log \log n) \).

Since \( \text{StUSPACE}(\log n) \) is a subclass of \( \text{DAuxPDA-TIME}(n^{O(1)}) \) we know that there are logtime \( \text{CROW} \)-algorithms for the elements of \( \text{StUSPACE}(\log n) \) [7]. To give better relative upper bounds on the complexity of \( \text{StUSPACE}(\log n) \) it is interesting to consider intermediate classes between \( \text{DSPACE}(\log n) \) and \( \text{DAuxPDA-TIME}(n^{O(1)}) \).

Trying to find these classes with sequential models could be difficult, since the usual restrictions of a pushdown store (e.g. the one-turn property, or using a counter instead of a push down) all collapse to logspace. Here, parallel machine models seem helpful, leading to two intermediate classes: the parallel pointer machine [6, 14] and the \( \text{OROW-PRAM} \) [20]. As a consequence of our main result we get \( \text{OROW} \)-algorithms for the elements of \( \text{StUSPACE}(\log n) \) taking time \( O(\log^2 n/ \log \log n) \). We are also able to show that all sets in \( \text{StUSPACE}(\log n) \) are accepted in logarithmic time on a parallel pointer machine. This latter containment is somewhat surprising, because there are characterizations in terms of parallel programs indicating that the class \( \text{PPM-TIME}(\log n) \) is rather close to \( \text{DSPACE}(\log n) \) [16].

## 2 Preliminaries

We assume the reader to be familiar with the basic notions of complexity theory (e.g. [10]). In addition, let \( \text{DTISP}(f, g) \) be the set of all languages accepted by \( O(g) \) space-bounded Turing machines in time \( O(f) \). \( \text{NTISP}(f, g) \) denotes the corresponding nondeterministic class.

We refer the reader to the survey article of Karp and Ramachandran [13] for coverage of the many varieties of parallel random access machines and their relationship to sequential classes. Let us remark here, that we deal in this paper only with algorithms and classes using \( \text{PRAMs} \) with a polynomial number of processors. The notion of a \( \text{CROW-PRAM} \) was introduced by Dymond and Ruzzo [7] and provides the tightest possible connections to deterministic machines [9].

\( \text{CROW-PRAMs} \) need only logarithmic time to recognize any given language in \( \text{DSPACE}(\log n) \); There are two important ways to restrict \( \text{CROW-PRAMs} \) and still maintain this property. One way is to restrict the concurrent read access to the global memory, which leads to the \( \text{OROW-PRAMs} \) of Rossmanith [20]. The other way is to restrict the arithmetical capabilities of the instruction set leading to \( r\text{CROW-PRAMs} \) and to parallel pointer machines [6, 14].