How to Implement First Order Formulas in Local Memory Machine Models

Elias Dahlhaus
Department of Computer Science, University of Bonn, Römerstrasse 164, D-5300 Bonn 1

Abstract. The paper indicates how the validity of a fixed first order formula in any finite structure can be checked by realistic local memory parallel computers.

1 Introduction.

There are a sample of results on the connections between the expressive power of formulas of certain type and the computational complexity of the classes of their finite structures [9, 12, 14, 17]. Such characterizations of complexity classes are of interest as they give syntactic descriptions of complexity classes. The practical use of such considerations could be that we get a tool to develop programming languages covering computational complexity classes like P, NP, logspace, and nondeterministic logspace. Recently Immerman [13] proved that, with auxiliary predicates arising from the input order, first order formulas cover those classes of finite models which can be recognized by a certain parallel random access machine model, the CRAM (concurrent random access machine), in constant time with a polynomial number of processors. The CRAM is an extension of the concurrent read concurrent write parallel random access machine (CRCW-PRAM) with the shift of a binary number as an additional in constant time operation. The degree of the polynomial assigning the number of processors depends on the number of variables of the first order formula [13]. Here we consider the parallel complexity of the evaluation of first order formulas in finite structures on more realistic local memory parallel computation models like the hypercube, the butterfly, and the shuffle exchange network (see for example [2]).

There are a couple of general techniques to translate parallel random access machine algorithms into local memory machine programs [18, 16, 1]. All these techniques have one of the following disadvantages: they use probabilistic elements or they are based on routing techniques on sorting networks. All deterministic sorting networks of logarithmic depth and size $O(n \log n)$ are based on expanders. The constants for the size and depth of expander networks are very high. In so far expander networks are no realistic sorting networks. The most common sorting networks are
the odd-even merge network and the bitonic sorting network [5]. Both networks can be simulated on a butterfly network or on a hypercube but they have a depth of $O(\log^2 n)$. Recently Cypher and Plaxton [8] developed a deterministic sorting algorithm on the hypercube with a linear number of processors and a time bound of $O(\log n \log \log n)$. This algorithm is only of theoretical interest because the constant in the time bound is very high.

One possibility to implement first order formulas in local memory machines like the hypercube, the butterfly, or the shuffle exchange network is the simulation of the corresponding CRAM-program by some general techniques as mentioned above. Here we shall show that the direct implementation into any of the well known local memory networks as the hypercube or the butterfly network or the shuffle exchange network is more efficient and can be done without probabilistic elements. We shall prove that first order formulas without function symbols can be evaluated by a hypercube, a shuffle exchange network, and by a butterfly in logarithmic time using polynomially many processors. The degree of the polynomial coincides with the number of variables of the first order formula (up to a logarithmic factor). Note that, in general, the processor number is optimal (up to a logarithmic factor), because the sequential time to evaluate a a first order formula with $k$ variables is $O(n^k)$. Moreover, also in case that parallel random access machines can be simulated in a logarithmic time loss by realistic local memory networks, the parallel techniques to evaluate first order formulas on finite models as they are presented in this paper may remain interesting, because they are quiet simple.

In section 2, the basic concepts and notions are presented. Section 3 introduces the parallel communication structures considered in this paper. In section 4, the main results are presented. Section 5 discusses the problem how to behave if the certain formula has function symbols. In the last section, we discuss some conclusions related to the results of this paper.

2 Basic Concepts

A similarity type or signature $L$ consists of a set $C$ of constant symbols, sets $R_i$ of relation symbols of arity $i$, and sets $F_i$ of function symbols of arity $i$.

A similarity type without function symbols is called a relational similarity type.

Without loss of generality, we assume that the domain of each finite structure is an initial segment $\{0, \ldots, n - 1\}$ of the natural numbers. We denote the set of structures of the similarity type $L$ with an initial segment of the natural numbers as domain by $Mod(L)$.

The first order language of the similarity type $L$ is denoted by $FO(L)$.

The set of variables of a first order formula $\phi$ is denoted by $Var(\phi)$. Note that the certain variable $x$ may appear in different places and is bounded by different quantifiers. Such variable is still counted only once.

Suppose the formula $\phi(x_1, \ldots, x_k) \in FO(L)$ has the free variables $x_1, \ldots, x_k$. Then, for each $M \in Mod(L)$, we set $F_\phi(M) = \{(u_1, \ldots, u_k)|M \text{ satisfies } \phi(u_1, \ldots, u_k)\}$. 