Distributed Deduction by Clause-Diffusion: the Aquarius Prover *

Maria Paola Bonacina and Jieh Hsiang

Department of Computer Science
SUNY at Stony Brook
Stony Brook, NY 11794-4400, USA
bonacina@loria.fr   hsiang@sbcς.sunysb.edu

Abstract. Aquarius is a distributed theorem prover for first order logic with equality, developed for a network of workstations. Given in input a theorem proving problem and the number n of active nodes, Aquarius creates n deductive processes, one on each workstation, which work cooperatively toward the solution of the problem. Aquarius realizes a specific variant of a general methodology for distributed deduction, which we have called deduction by Clause-Diffusion and described in full in [6]. The subdivision of the work among the processes, their activities and their cooperation are defined by the Clause-Diffusion method. Aquarius incorporates the sequential theorem prover Otter, in such a way that Aquarius implements the parallelization, according to the Clause-Diffusion methodology, of all the strategies provided in Otter.

In this paper we give first an outline of the Clause-Diffusion methodology. Next, we consider in more detail the problem of distributed global contraction, e.g. normalization with respect to a distributed data base. The Clause-Diffusion methodology comprises a number of schemes for performing distributed global contraction, which avoid the backward contraction bottleneck of purely shared memory approaches to parallel deduction. Then, we describe Aquarius, its features and we analyze some of the experiments conducted so far. We conclude with some comparison and discussion.

1 Introduction

In this paper we describe the Clause-Diffusion methodology for distributed theorem proving, its implementation in the theorem prover Aquarius and we analyze the performances of Aquarius on some experiments.

A theorem proving problem consists in deciding, given a set of clauses S and a clause ϕ, whether ϕ is a theorem of S. A theorem proving strategy C is specified by a set of inference rules I and a search plan Σ. The inference rules

* Research supported in part by grant CCR-8901322, funded by the National Science Foundation. The first author is also supported by a fellowship of Università degli Studi di Milano, Italy. First author's current address: INRIA-Lorraine, 615 Rue du Jardin Botanique, B.P. 101, 54602 Villers-les-Nancy, France.
can be further separated into two classes. The \textit{expansion inference rules}, such as resolution and paramodulation, derive new clauses from existing ones and add them to the data base. The \textit{contraction inference rules}, such as simplification and subsumption, delete clauses or replace them by smaller ones. The search plan $\Sigma$ chooses the inference rule and the premises for each step, so that the repeated application of $\Sigma$ and $I$ generates a derivation. A derivation is successful if it reaches a solution of the input problem. If the strategy is \textit{complete}, the derivation is guaranteed to succeed whenever the input goal is indeed a theorem. In practice, however, a derivation by a complete strategy may fail to prove a theorem, because it generates so many clauses that it exhausts the available memory before succeeding. In other words, it generates too large a portion of the \textit{search space} of the problem. \textit{Contraction-based strategies} try to reduce the incidence of such failures by applying eagerly powerful contraction rules to keep the data base, and thus the search space, as reduced as possible. These strategies, as implemented for instance in the provers Otter [20], RRL [16] and SBR3 [1], have obtained very encouraging results.

In this paper we present a methodology for parallelizing contraction-based deduction strategies. The main feature of contraction-based strategies is that existing data may be deleted or replaced by others through contraction. For instance, an equation may be reduced to another equation via rewriting. Although such a behaviour is the main reason why contraction-based strategies are effective, it is also the major source of difficulty in parallelization. To illustrate this point, we consider the parallelization of Prolog technology theorem proving (PTTP) methods. In goal-reduction methods such as PTTP, the set of axioms remains static during the course of the derivation. Thus, it is possible to pre-process all the axioms into elaborate data structures before the derivation starts. Such structures are used to exploit parallelism of different granularities. The cost of building them is limited to the pre-processing phase. Contraction-based strategies, on the other hand, are not likely to take advantage of such approaches, because axioms will be added and deleted during the derivation, so that pre-processing is not sufficient.

The basic idea of the Clause-Diffusion methodology, which we present here, is to parallelize a strategy \textit{at the search level}, by partitioning the search space among many concurrent deductive processes, which search in parallel for a solution. As soon as one of them succeeds, the whole distributed derivation succeeds. The deductive processes are asynchronous and work in a largely independent fashion: each process has its own local data base, constructs its own derivation and interacts with the others through \textit{message-passing}.

The Clause-Diffusion approach has a few features which we consider as new:

- It is a general methodology intended for implementing contraction-based strategies in distributed environments.
- The problem of keeping data inter-contracted is dealt with through a notion of \textit{image set} – an approximation of the global data base. This avoids the difficulty of the \textit{backward contraction bottleneck} which often occurs in shared memory implementation of contraction-based strategies [18, 27].