ELSEVIER

Expert Systems with Applications 29 (2005) 554–565

Expert Systems with Applications

www.elsevier.com/locate/eswa

A multi-agent framework for distributed theorem proving

Chih-Hung Wu*

Department of Electrical Engineering, National University of Kaohsiung, 700 Kaohsiung University Road, Kaohsiung 811, Taiwan

Abstract

In this paper, we explore the possibility of distributed theorem proving using agent-based technologies. We investigate the hyper-linking theorem proving and find out its components which can be performed independently by agents. A multi-agent framework is proposed, wherein distributed theorem proving is achieved by the collaboration of multi-agents each of which performs part of the hyper-linking strategy for completing the proof of the given problems. In this framework, agents communicate with each other via KQML-based messages. Additionally, several system agents are designed for monitoring the traffic and performance of the framework and sharing information. The architecture and the design concepts are addressed. Experimental results show that the proposed framework can effectively perform distributed theorem proving on the Internet.

Keywords: Intelligent agents; Multi-agent system; KQML; Theorem proving; The hyper-linking strategy; First-order logic

1. Introduction

For past decades, scientists have been interested in automated deduction systems which can be applied to expert systems, planning, common sense reasoning, deductive databases, hardware/software verification, and many other areas. The so-called theorem proving is a core-technique in a deduction system, which tries to 'prove' the formal correctness of a set of clauses describing the underlying problem. Unfortunately, many problems in theorem proving are NP-complete, even NP-hard (Chang & Lee, 1973); and it happens that finding proofs of a hard problem may require tens of hours of execution on workstations due to the huge search space. This motivates many researchers to explore parallel or distributed computation for high-performance automated deduction systems.

Several high-performance parallel deduction systems have been proposed and implemented, such as those discussed in Section 5. Parallelizing a sequential system can be studied with respect to fine-grain coarse-grain strategies by executing machine instructions or sub-processes of the system in parallel, respectively. Experimental results show that many computational efforts in finer-grain parallelism are not directive and futile. Besides, implementing finer-grain parallelism is costly since it usually needs (1) expensive hardware such as computer systems with a number of processing units and high-speed memory modules and (2) a lot of human-hours for developing and testing parallel algorithms and (3) efficient, but usually complicated, communication protocols for efficient data exchange. Attention has been shifted to coarser-grain parallelism for deduction systems (Bonacina & Hsiang, 1994).

Fortunately, with standard connection protocols like TCP/IP, there are a number of inexpensive, heterogeneous, computers connected on the Internet which seemingly form an ideal distributed environment for general-purposed computation as well as for distributed theorem proving. However, if Internet-based computation is applied to distributed theorem proving, the following issues have to carefully considered.

(1) Task Partitioning and Load Balancing. Deductive systems are computationally intensive. Distributed processing usually divides a complicated task into several sub-tasks each of which is performed on a computing machine and reassembles the results returned from each machine. If the main task is improperly partitioned, the loads of computing machines are unbalanced. The grainity to be distributed processed should be carefully designed.

* Corresponding author. Tel.: +75 919 446; fax: +75 919 374.
E-mail address: johnw@nuk.edu.tw

0957-4174/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2005.04.013

(2) Communication. Evidently, even if tasks are properly partitioned, they unavoidably need to stop for communicating with each other for data exchange for completing the main task. Clearly, volume of communication degrades the performance of the whole system. A clearly known fact is that the reliability of the Internet is less than well-designed and tested multiprocessor computers. Connecting to such heterogeneous computing machines may encounter unexpected delays of traffics. A standard for communication is needed for fast exchange of information.

(3) Control and Monitoring. Deductive systems apply different directive parameters which may be set heuristically. Such parameters need to be carefully adjusted. Also, deductive problem-solving is goal-oriented, no matter it is done in a centralized or distributed manner; there should be a means to examine if the goal is achieved. A management mechanism for monitoring the progress and reliability of all computing machines is essential.

Recently, Internet-based collaboration of multi-agents is an interesting research topics and has been receiving a lot of attentions (Hendler, 2001; Maturana et al., 2004; Milano & Roli, 2004). Software agents are computer programs capable of flexible, autonomous, and goal-oriented actions. In their most complex form, agents may persist over time, are capable of timely internal context dependent reaction to sensed events, plan and initiate unique series of actions to achieve stated goals, and communicate with other agents (or people) for completing the given goal. It has been suggested that multi-agent systems are specially adequate for the solution of problems with a dynamic, uncertain and distributed nature (Aldea et al., 2001). In this paper, we propose a framework of multi-agent system for distributed theorem proving. We demonstrate our idea using the hyper-linking proof procedure (Lee & Plaisted, 1992) which is a refutational deduction method in first-order logic. In this framework, distributed theorem proving is achieved in clause-level by the collaboration of multi-agents. Agents solve the given problem and exchange information for sharing data and adjusting directive parameters for proving problems. A simple, KQML-based (Finin, Labrou, & Mayfield, 1997), scheme has been designed for exchange of information among agents in a peer-to-peer style. The advantages of use of this framework include lower cost and higher expandability for distributed theorem proving, usefulness for solving large and complex problems, and being easily to be applied to other theorem proving techniques.

We describe the design of the framework and the role of each agent in this paper. The interaction and the collaboration of agents are also discussed. The rest of the paper is organized as follows. Section 2 gives a brief description of the hyper-linking strategy and discusses the tasks which can be partitioned and collaborated for distributed processing. In Section 3, the architecture of the multi-agent framework and the design of each agent are described. Section 4 demonstrates the system and gives some preliminary results. Related researches are discussed in Section 5 followed by a summary of this study in Section 5.

2. Background

2.1. The hyper-linking theorem proving

The hyper-linking proof procedure (HLPP) was proposed to eliminate the duplication of clauses occurring in many deduction methods and has been proved to be efficient (Lee & Plaisted, 1992) compared with widely known systems such as OTTER (McCune, 1994), *sprfn* (Plaisted, 1988), and PTTP (Stickel, 1988) in proving some problems in first-order logic. HLPP is a refutational clause-form deduction method, consisting of the following two processes: hyper-instance generation (HIG) and propositional satisfiability test (PSAT). Below we give a brief introduction to HLPP.

Suppose R is a set of clauses in first-order logic. HIG produces hyper-instances by performing unification between literals of clauses in R to instantiate these clauses. If $C=L_1\vee\ldots\vee L_m$ is a clause in R, and $M_1,\ldots,M_m$ are literals from the clauses in R, with variables renamed to avoid conflicts, and Θ is a most general unifier such that $L_i\Theta$ and $M_i\Theta$ are complementary for all i, $1\leq i\leq m$, then $C\Theta$ is called a *hyper-instance* of C. The set $\{(L_1,M_1),\ldots,(L_m,M_m)\}$ is called a *hyper-link*. We call C a *nucleus* and M_i the *electrons* of C. We denote by $H(R)$ the set of hyper-instances of all the clauses in R and by $Gr(R)$ the ground set of R where all variables in R are replaced by the same constant symbol \$. Let S_0 be the set S of input clauses and S_i be $S_{i-1}\cup H(S_{i-1})$, $i\geq 1$. HIG takes clauses in S_i as nuclei and generates all hyper-instances, $H(S_i)$. Then PSAT tries to decide the satisfiability of $Gr(S_i\cup H(S_i))$. If $Gr(S_i\cup H(S_i))$ is unsatisfiable, then S is unsatisfiable and we are done; if all clauses in $H(S_i)$ are contained in S_i, i.e. $S_i=S_i\cup H(S_i)$, then S is satisfiable and we are also done. Otherwise, one more round of HIG and PSAT is performed on S_{i+1}. HLPP is a saturation-based method, i.e. no clauses in S_{i+1} can be considered as a nucleus before all the hyper-instances of S_i have been generated.

The efficiency of the proof procedure can be improved by filtering hyper-instances through unit-simplification wherein hyper-instances containing specific unit-literals are reduced or removed from the database. A unit-literal is a literal from a clause containing only one literal, which usually describes a fact given by the users or derived from the system. Similar to that in the Davis-Putnam procedure (Davis & Putnam, 1960), two simplification rules, *unit literal deletion* and *unit subsumption*, are employed in unit-simplification. If all the literals of a clause are removed,