Development of the Methods and Tools for Mathematically Correct Logic Programming of Internet Agents[1]

A. A. Morozov and Yu. V. Obukhov

Institute of Radio Engineering and Electronics, Russian Academy of Sciences, ul. Mokhovaya 11, Moscow, 125009 Russia

e-mail: {obukhov, morozov}@mail.cplire.ru

Abstract—In this paper, we consider the development of a mathematical apparatus for logic programming of intelligent agents for searching, recognizing, collecting, and analyzing information on the Internet. The problem of ensuring soundness and completeness of logical inference in the dynamic environment of the Internet under the conditions of permanent update and revision of information is discussed. For logic programming of the Internet agents, we propose a new mathematical apparatus based on the principle of repeatedly proven subgoals.

INTRODUCTION

Internet agents are programs that automate retrieval, recognition, extraction, and analysis of information on the Internet; these factors take account of the needs and goals of an individual user (or a group of users). They differ from the widely used retrieval systems in the following:

(1) agents can autonomously operate during long periods of time (days, weeks, or more);

(2) once created, an agent can be used many times (like any other program), whereas a request to a retrieval system causes a single operation of data acquisition.

To date, there has been no generally accepted definition of intelligent agents. Usually, however, programs that are autonomous, reactive (i.e., react to external stimuli), proactive (e.g., capable of planning further actions), and exhibit a social behavior (if there are many agents in a system) are called intelligent agents [1, 2].

One of the most interesting and promising approaches to programming Internet agents is logic programming of agents [1, 3]. This approach has good prospects, because the ideology and principles of logic programming are very convenient for searching, recognition, and analysis of unstructured, poorly structured, and hypertext information [7, 9]. Many ideas and methods of logic programming of Internet agents based on various modifications and extensions of Prolog and nonclassical logic (linear, modal, F-logic, etc.) were developed during the last decade [1, 3, 7]. Nevertheless, there is still no mathematical apparatus providing sound and complete operation of logic programs in the dynamic Internet environment (i.e., under conditions of permanent update and revision of information). To solve this problem, we have created a mathematical apparatus based on the principle of repeatedly proven subgoals.

THE IDEA OF MATHEMATICAL APPARATUS

Our mathematical apparatus for logic programming of Internet agents includes:

(1) A model of intelligent agents that operate in a dynamic environment;

(2) A declarative (model-theoretic) semantics of agents;

(3) Control strategies for executing logic programs (Internet agents) that are sound and (under some conditions) complete with respect to the model-theoretic semantics of these agents.

Within the framework of our model of intelligent agents, an Internet agent (or a group of interacting Internet agents) is a logic program controlled by a special strategy which implements the so-called repeated proving of subgoals.

The idea of repeated proving consists in dividing the program into separate subgoals (called logical actors) [4–9] that have the following properties:

(1) Common variables are the single channel of data exchange between the actors.

(2) The proving of separate actors can be fulfilled independently in arbitrary order.

(3) One can cancel the results of proving of any actor while keeping all other subgoals of the program.

After canceling the results of the proving of the actor, its proving can be repeated. Thus, one can implement a modification of reasoning. The logical inference can be partially modified. The results and the consecution of reasoning itself can be partly modified in the

[1] This work was supported by the Russian Foundation for Basic Research, project no. 00-01-00560.

Received October 29, 2002

process and after the logic inference. This makes it possible to eliminate the contradictions between the results of the logical reasoning and new information.

We use the developed object-oriented syntactical means (classes, inheritance, etc.) for structuring a search space of logic programs [4–6, 9].

SYSTEMS OF LOGICAL ACTORS AND DECLARATIVE SEMANTICS OF AGENTS

The most complicated and interesting problem to be solved for implementing the idea of logical actors and repeated proving is the development of control strategies supporting repeated proving which are sound and (if possible) complete. We have developed several control strategies supporting repeated proving.

One of the first control strategies was created for the execution of sequential logic programs with logical actors. This strategy was described in detail in [5], where the theorems about soundness and completeness of this strategy were proved for logic programs without looping.

Further experiments on visual logic programming of intelligent agents have shown that it is expedient to develop more complex concurrent control strategies as well. To the present day, we have expanded our computing model by introducing concurrent processes. Each process is a system of logical actors. The processes interact by using asynchronous messages of two kinds [9]:

(1) the so-called direct messages (intuitively, they correspond to asynchronous call of the predicates in one concurrent process from another one);

(2) flow messages (corresponding to data transmission through the common variables of processes).

The composition of messages of these two kinds helps us to describe the complex behavior of agents (or systems of interacting agents) without means of synchronization of concurrent processes. One can find a detailed description of our methods of asynchronous information exchange and process states in [4, 9].

The main advantage of our expanded computing model is that it provides a declarative (model-theoretic) semantics of concurrent logic programs. At the same time, the strategy of concurrent programs with message exchanging is not complete with respect to their model-theoretic semantics. A strict definition of the model-theoretic semantics of concurrent logical actors and a theorem about the soundness of the actor mechanism are given in [9].

The conditions under which the parallel logical program is not only sound but also complete with respect to its declarative semantics are a separate subject for investigation. Obviously, this can be provided only if rather strict limitations on the properties of separate processes and on the system of processes as a whole are fulfilled.

(1) Each separate process (as a separate logic program that computes some data) must be complete with respect to its declarative semantics. In particular, the absence of process cycling should be guaranteed.

(2) The procedures that compute data transmitted from one process to another must be deterministic in order that the completeness of logic inference is not lost due to the impossibility of backtracking from the receiving process to the transmitting one.

(3) The information transmission between the processes should be strictly one-directional. Only flow messages should be used (this condition can be, in principle, softened). The system of interacting processes should have no feedback.

The systems of interacting processes complying with these strict conditions are of great practical importance, because they describe the flow processing of data received by a system of intelligent agents from outside. The soundness and completeness properties ensure the exhaustive search, i.e., guarantee all solutions that fulfill the given logical conditions.

IMPLEMENTATION AND PRACTICAL USE

We have created an object-oriented logic language Actor Prolog on the basis of our mathematical apparatus (the definition of the language, including all new means, is available at our Website [4]). We have also introduced some special means that support programming of Internet agents in recent versions of the language:

(1) There are predefined classes implementing HTTP and FTP protocols.

(2) There are means for visual programming based on translation of Structured Analysis and Design Technique (SADT) diagrams into Actor Prolog [9].

(3) There are syntactical features supporting component-oriented programming.

Now, we have a working version of a system of logic programming of intelligent Internet agents on the basis of the developed mathematical apparatus and Actor Prolog player.

PERSISTENT AGENTS

All programs written in Actor Prolog are long-lived or persistent. A user can at any moment save the state of the operating program into a file and then restart it from that point. Actor Prolog saves states of all objects of a program, including the contents of text windows and current positions of open files. This feature is of great importance for Internet agents, because they operate during long periods of time. So, they can restore their states in cases of hardware or software malfunctions in the computer system. Moreover, a user can save the states of his/her agents, switch off the computer, and go home at any time. One can also transfer an agent from one computer to another via a regular floppy disk.

COMPARISON WITH OTHER APPROACHES

The main advantage of our model and methods of programming Internet agents is the use of control strategies that support repetitively proving subgoals. The method of repetitive proving provides (classical) soundness of logic programs operating in a dynamic environment as opposed to other approaches based on Prolog extensions (e.g., LogicWeb, Minerva, and DLP), constraint logic programming (e.g., Oz language, etc.), and nonclassical logic (e.g., W-ACE and Lygon). In fact, the developed method of repetitively proving subgoals can be generalized and used for implementing constraint logic programming and nonclassical logic programming. However, all those types of programming themselves do not provide the classical model-theoretic semantics of the agents operating in the dynamic environment.

We have implemented modifying reasoning, staying within the framework of classical first-order logic. Thus, our approach is free from the disadvantages of non-monotonic logic systems, such as the absence of general validity (in a classical sense) of the inferred formulas, the dependence of the result on the order of the rules of inference, and the high probability of cycling in logic programs.

CONCLUSIONS

A mathematical apparatus of logic programming of intelligent agents implementing information retrieval and recognition in the highly structured dynamic environment of the Internet is developed. It is based on repetitively proving subgoals, which allows us to modify logical reasoning during and after operation of the logic program by adjusting it to the newly arriving information from the outside.

On the basis of the developed apparatus of the modifiable logical reasoning, we designed a parallel object-oriented logical language Actor Prolog which ensures the soundness of logic programs (intelligent agents) operating under the conditions of permanent change and updating of information. Actor Prolog supports visual and component-oriented agent programming, as well as persistent agents.

ACKNOWLEDGMENTS

We are grateful to Academician Yu. I. Zhuravlev and Professor V. A. Zakharov for valuable discussions of the problem of describing the declarative semantics of multiagent systems.

REFERENCES

1. Sadri, F. and Toni, F., Computational Logic and Multiagent Systems: A Roadmap, 1999 (available at http://citeseer.nj.nec.com/sadri99computational.html).
2. Huang, Z., Eliens, A., van Ballegooij, A., de Bra, P., A Taxonomy of Web Agents, *Proc. of DEXA Workshop*, 2000, pp. 765–769. (available at http://citeseer.nj.nec.com/408821.html).
3. Davison, A., Logic Programming Languages for the Internet, in *Invited Submission for Computational Logic: From Logic Programming into the Future*, Kakas, A. and Sadri, F. (Eds.), Springer, 2001 (available at http://five-dots.coe.psu.ac.th/~ad/papers/summBob.ps.gz).
4. Morozov, A.A. and Obukhov, Yu.V., Actor Prolog: Definition of Programming Language. *Preprint of IRE RAS*, Moscow, 1996, no. 2(613), p. 57, (the recent version of the language definition is available at http://www.cplire.ru/Lab144/index.html).
5. Morozov, A.A., Logic Analysis of Functional Diagrams in the Interactive Design of Information Systems, *Cand. Sci. Dissertation*, Moscow, 1998, p. 199 (available at http://www.cplire.ru/Lab144/auto.html).
6. Morozov, A.A., Actor Prolog: An Object-Oriented Language with the Classical Declarative Semantics, *Proc. IDL'99 Workshop*, Paris, 1999 (available at http://www.cplire.ru/Lab144/paris.pdf).
7. Morozov, A.A., Obukhov, Yu.V., and Gulyaev, Yu.V., On the Problem of Using Logic Object-Oriented Programming in the World Wide Web, *Proc. of the Special Russian Session "The Internet Developments in Russia". First IEEE/Popov Workshop on Internet Technologies and Services*, Moscow, 1999. pp. 54–59 (available at http://www.cplire.ru/Lab144/internet.pdf).
8. Morozov, A.A. and Obukhov, Yu.V., On the Problem of Logical Recognition in the Dynamic Internet Environment, *Pattern Recognit. Image Anal.*, 2001, vol. 11, no. 2., pp. 454–457 (available at http://www.cplire.ru/Lab144/pria5.pdf).
9. Morozov, A.A., and Obukhov, Yu.V., An Approach to Logic Programming of Intelligent Agents for Searching and Recognizing Information on the Internet, *Pattern Recognit. Image Anal.*, 2001, vol. 11, no. 3, pp. 570–582 (available at http://www.cplire.ru/Lab144/pria570m.pdf).