A List Processing Language TAO with Multiple Programming Paradigms

Ikuo TAKEUCHI, Hiroshi OKUNO and Nobuyasu OHSATO
Electrical Communication Laboratories,
Nippon Telegraph and Telephone Corporation,
3-9-11 Midoriho, Musashino, Tokyo 180.

Received 30 October 1985
Revised manuscript received 31 August 1986

Abstract This paper describes an interpreter-centered list processing language TAO which supports the logic programming paradigm and the object-oriented programming paradigm together with the conventional procedural programming paradigm in the framework of the Lisp language. TAO allows the user to mix these programming paradigms in solving complicated and multifaceted AI problems. The fundamentals of these programming paradigms, namely, unification, message passing and function call can nest each other in an expression. Thus, the user can use the result of a function call or a message passing in a unification straightforwardly and vice versa. TAO also supports the concurrent programming. The implementation of the TAO interpreter on a Lisp machine called ELIS achieves a remarkable efficiency.

Keywords: Multiple Programming Paradigms, List Processing Language, Object-oriented Programming, Logic Programming

What is said TAO is not the true TAO — Lao-Tse

§1 Introduction

AI programming has been proved to be so complicated and multifaceted that no single programming paradigm seems to be able to cope with it sufficiently. AI languages come to be desired to involve many functions or programming paradigms. A few languages and systems are proposed along this line: LOOPS\(^2\) and ESP,\(^3\) for examples. TAO is a newcomer to the family of these multiple programming paradigm languages.

Basically, TAO is a Lisp dialect. However, it assimilates the essence of the object-oriented programming and the logic programming into the heart of the
Lisp world. That is, fundamental computation mechanisms for these programming paradigms are implemented and united into a single evaluation kernel of the interpreter, called eval. (See Appendix for the brief description of eval.) Roughly speaking, TAO provides the s-expression forms with the semantics of lambda calculus, Horn clauses, message passing, data abstraction, and conventional "load-and-store" computation based on the address concept. The user can write programs, mixing those at the expression level, not only at the program module level. In other words, TAO accommodates semantics of Lisp, Prolog, Smalltalk and Fortran in a harmonious way. We believe that TAO is the first attempt to realize this kind of intimate mixture of these programming paradigms.

The word "harmonious" means another thing. The performance of these programming paradigms is well-balanced in TAO. We believe that each of four programming paradigms has been implemented with almost same or comparable performance, with respect to the execution speed, memory consumption, functionality and programming ease. The user can select and combine these paradigms adequately for his purpose. The user need not worry about how to tune his programming style to the performance profile.

TAO is an interpreter-centered language in the sense that its semantics makes the interpreter as fast as possible and makes the compiler merely an optimizer for the interpreter. Thus, the user can enjoy the fully interactive programming even when he is making a large scale application program. At present, TAO has only the interpreter on a Lisp machine called ELIS, and the compiler is under development. Hence, the performance figures in this paper is that of the interpreter unless otherwise mentioned. However, the execution speed of the interpreter is remarkably high. It is comparable with compilers of other systems such as MacLisp on DECsystem-2060 if running programs are large enough.

§2 Design of the Programming Language TAO

TAO is a programming language designed mainly to cope with complicated multifaceted AI problems. However, from another viewpoint, it can be conceived as a versatile programming language that can cope with many other kinds of information processing problems. Indeed, AI is one of the ultimate problems of general information processing.

In this section, we discuss how a multiple programming paradigm language TAO is designed on the ELIS Lisp machine, aiming at the maximum functionality and the minimum runtime cost, and also we describe the overview of TAO.

2.1 Design Objectives

When we are required to write programs, using different programming paradigms, it would be better for us to use a single programming language with