Research on Decompiling Technology

Liu Zongtian (刘宗田) and Chen Fuan (陈福安)

Microcomputer Institute, Hefei University of Technology, Hefei 230009
Received November 6, 1991; revised May 6, 1994.

Abstract
Decompiling, as a means of analysing and understanding software, has great practical value. This paper presents a kind of decompiling method offered by the authors, in which the techniques of library-function pattern recognition, intermediate language, symbolic execution, rule-based data type recovery, program transformation, and knowledge engineering are separately applied to different phases of decompiling. Then it is discussed that the techniques of developing expert systems are adopted to build a decompiling system shell independent of the knowledge of language and program running environment. The shell will become a real decompiler, as long as the new knowledge of application environment is interactively acquired.

Keywords: Decompiler, pattern recognition, intermediate language, symbolic execution, data type recovery, program transformation, knowledge-based system.

1 Introduction
A large number of programs have been produced since the creation of computer. They are an embodiment of the human wisdom and a very important wealth in the treasure-house of science and culture. Reading and analysing them are not only an effective way to acquire the programming experience and enhance the programmer's quality but also a basic means to maintain, transplant, and improve the programs.

Although great number of programs are written in high-level programming languages at present, most of them are usually offered to the users in the form of machine code because of the memory limit and the technical security. Sometimes, a firm may lose source code of a program developed by its own people for some reasons. For example, source code may be taken away by some developer who left suddenly or it may be destroyed by some mistakes.

Reading the machine code is very difficult so that a software tool is expected to translate the machine code into the high-level language or the specification. That tool is decompiler, or anti-decompiler. Decompiling, as a means of analysing and understanding software, is an important part of the software reverse engineering and has great practical value.

Like other reverse engineering tools, decompiler is also confronted with the question whether its use is legal. The question has been argued for years. Although no agreement has been reached, most of lawyers and governments have accepted concerted opinion at the points below[1,2]:

(1) A tool or a technology, if it may be of benefit to human advancing, ought not to be banned by law.

(2) The problem about legality depends on the purpose of using the tool or technology, instead of depending on the tool or the technology itself.

(3) At least, decompiler can legally be used for the purposes:
   a) recovering source code of a program developed by yourselves;
b) analyzing software developed by other firm for learning and researching the ideas, algorithms, techniques, logic and functional elements of the software;

c) making modifications for fixing a bug, enhancing the software so that it can better perform its function;

d) developing a new software which only other software's ideas, algorithms and logic are incorporated into.

Research on the decompiling system was conducted abroad in the 1970's, then it gradually ceased. The reasons are as follows: (1) the decompiling techniques were too difficult to be developed to a practical system at that time; (2) reading and understanding the other persons' programs are also a hard work, even if the programs are in some form of high-level language. Many programmers prefer to design programs by themselves rather than read the existing and similar programs; (3) the software developing methods by transplanting and improving the existing programs did not attract enough attention at that time.

Many organizations in China have been doing research on the decompiling system one after another since the middle of 1980's, such as the 68000C system on DUAL 68000[3] and the 8086 C system on IBM-PC[4] developed by the authors, and the C decompiler on VAX by Shanghai Jiaotong University, etc.

The research on the decompiling system in other countries has gradually increased in recent years, and the artificial intelligence technology begins to be applied to developing systems[5].

The decompiling process may start from machine code without symbolic message such as names of library-functions, global-variables, and local variables, or from machine code with symbolic message, or from intermediate code such as the postfix Polish or the P code. The first form is most difficult and useful. This paper presents a no-symbol-decompiling method which adopts the techniques of library-function pattern recognition, intermediate language, symbolic execution, rule-based data type recovery, program transformation, and knowledge engineering. The decompiling process which translates the machine code into high-level language or the specification generally goes through the following phases (shown in Fig. 1).

![Diagram](image)

**Fig. 1. Phases of decompiling.**

The techniques and methods applied to different phases of decompiling are separately described in the following sections. It is also discussed that the techniques of developing expert systems are adopted to build a decompiling system shell independent of the knowledge of language and program running environment.