Reusable Parser Generation from Open Source Compilers

Kazuaki Maeda

Chubu University
1200 Matsumoto, Kasugai, Aichi 487-8501, JAPAN
kaz@acm.org

Many Open Source Software (called OSS) projects have been proposed and many software developers have contributed to develop software by OSS style. In the OSS development style, the source code is opened to the public and checked by the distributed software developers to improve the quality. The source code is, however, not effectively used to improve the productivity of other software development. This paper describes reusable parser generation from the source code of popular open source compilers.

In construction of code analyzers or reverse engineering tools, the parser development is a time-consuming task. To improve the productivity of the task, a renewal parser generator MJay was developed. MJay generates grammar definitions and some utility programs. It is useful to construct software tools to analyze source code.

Based on my experiences to construct software tools to generate UML diagrams from source code, there are three approaches to develop the parser.

1. To develop a parser from scratch by reading the programming language specification.
   It takes about one week to develop the parser from scratch to the best of my knowledge. There are some cases where it takes more than one week to develop it with high quality because the specification of recently popular programming languages is very complex. It is too long to catch up with the short-term development in the current situation as agile software development grows in popularity.
2. To get grammar definitions from major web sites, or find them using web search engines.
   There are some web sites including collections of public grammar definitions[1]. The collections in the web sites are very useful, but many public grammars contain errors and they provide no sufficient guarantee that they are strictly correct. As a result of this, we must debug them to improve the quality by ourselves with spending much time.
3. To extract source code of the parser from open source compilers.
   There are free open source compilers available with high quality. One of the famous compilers is GNU compiler collection[2]. The other is Mono C# compiler which is an open source implementation of .NET development environment available on major operating systems (e.g. Linux, Mac OS X,

Please use the following format when citing this chapter:

Maeda, K., 2006, in IFIP International Federation for Information Processing, Volume 203, Open Source Systems, eds. Damiani, E., Fitzgerald, B., Scacchi, W., Scotto, M., Succi, G., (Boston: Springer), pp. 343-344

Solaris and Windows) [3]. These compilers, however, were developed with only consideration for generating object code from source code. It is difficult to extract only the parser to reuse for another purpose because it is tightly coupled with other modules in the compiler.

This paper proposes the other approach, that is, to replace the parser generator with a renewal parser generator MJay. If we develop a parser for C#, we can reuse Mono C# compiler[3]. The parser in Mono C# compiler is developed using a parser generator Jay. After the replacement of Jay with MJay, MJay generates grammar definitions for a reusable parser in addition with a commonly used LALR parser. As a result of this, the parser in Mono C# compiler is opened and we can construct software tools quickly.

The development process is the following;

1. MJay reads the grammar definition G, and it generates the parser P1 of the usual C# compiler written in C#, the grammar definition H and some utility programs for the reusable parser.
2. P1 and the related files are compiled, and the special Mono C# compiler is built. The compiler reads C# source code and generates parser behavior in addition with the object code. The parser behavior consists of primitive actions for a typical LALR parser, for instance, shift, reduce, et al.
3. Jay reads the grammar definition H and generates the reusable parser P2 written in C#.
4. P2 and the related files are compiled by the usual Mono C# compiler, and a software tool is built. The reusable parser P2 reads the parser behavior and it takes the same sequence of actions as the parser P1 does.

In summary, this paper describes the motivation and the idea about reusable parser generation from the source code of popular open source compilers using the renewal parser generator MJay. It is based on my hard experiences of constructing reverse engineering tools, by oneself, which extract design information and draw diagrams (e.g. class diagram, communication diagram, et al.) from source code. It took a few weeks to construct it according to traditional parser development. MJay was developed to help me build the parser as soon as possible.

Now another reverse engineering tool for Visual Basic is under construction. It took just only two hours to develop the parser using MJay. I believe that MJay becomes an important tool to construct programming tools using open source compilers.

References

1. Grammar List, *http://www.antlr.org/grammar/list.*
2. Free Software Foundation, GCC Home Page, *http://gcc.gnu.org/* .
3. Main Page - Mono, *http://www.mono-project.com/Main_Page* .
4. jay Homepage, *http://www.informatik.uni-osnabrueck.de/alumni/bernd/jay/* .