Development of a Compiler Construction Toolkit for an Introductory Course in Compiling

Dr. G. F. GANCHEV
Computer Science Department
University of Botswana, P. Bag 0022, Gaborone, Botswana
E-Mail: ganchev@noka.ub.bw
Facsimile: (267) 356591

Abstract: The compiler construction toolkit under development in the University of Botswana (UB) is intended to allow the students build rapidly components of compilers for languages that they define. It consists of a case-type scanner generator, a top-down and a bottom-up parser generators, a stack machine emulator, and utility programs. The two parsing strategies adopted are mixed precedence (2,1;1,1) and LL(1). The semantic routines are built by the students in an interactive regime. The toolkit allows experimenting with both one-pass and multi-pass compilation to a stack machine language. This language is always interpreted. The utility programs contain symbol-table maintenance and abstract syntax tree maintenance routines. Other routines (e.g. for code optimization) may be added after gaining more experience with the use of the tool.

The compiler construction course at UB is offered to both Mathematics and Computer Science students who do not have prior knowledge of the topic. The aim of the course is to explain the concepts from a practical perspective without ignoring the grammar theoretical issues of compiler design. The students have different backgrounds. Some of them are not familiar with formal languages and automata theory. The syllabus provides full coverage of the basics while addressing in more detail the issues of syntax directed translation.

The UB toolkit differs from other similar systems by the intentional simplicity of the concepts used. By the experience of the author novices with little mathematical background have difficulties in understanding such topics as automata-based models, attribute-value flow, semantic specifications. In contrast, more mechanistic concepts are easily understood. For example, students understand better the construction of a case-type scanner than building a finite state automata. They understand more easily the construction of precedence-based shift-reduce parsers than LR(k) parsers. They prefer building translation schemes to defining semantic functions.

Another objective in the development is the students' involvement. The system is being built entirely in the form of coursework assignments and projects. Most of the students taking the course have gone through a thorough software engineering course just a semester earlier. Taking part in the toolkit development gives them the opportunity to get involved in a quite large project.

Keywords: Syntax Directed Translation, Scanning, Automata Theory, Parsing, LL(1) Grammar, MSP Grammar, Context Analysis, Code Generation, Abstract Syntax Tree, Stack Machine