

Visual Parse++ Product Review

Jerry Fitzpatrick

Visual Parse++ is an innovative lexical analyzer and parser generator that has a GUI interface for creating and debugging a grammar file. Although the visual interface sets it apart from similar tools, Visual Parse++ is by no means a lightweight application.

VP++ is ideal for anyone who has wanted create their own scripting or programming language, or analyze an existing language in some new way. In fact, I have used VP++ extensively for parsing C, C++ and Java programs to determine their ABC counts^{1,2}. In this regard, I've found VP++ to be a powerful and useful Windows application.

The mainstay of parsing tools are the UNIX utilities `lex` and `yacc` and their derivatives. VP++ combines the capabilities of these separate utilities into one application, and uses a single grammar file to describe both the lexical tokens and production rules. It should be noted that the syntax of the grammar file is very different than that used by `lex` or `yacc`, and can take some getting used to.

One of the neat features of the VP++ lexical analyzer is its stack-based expression lists, which allow nested expressions to be handled more easily. The classic use of this feature is to recognize and gobble up comments or string literals, then return to the main stream of lexing. This capability is very powerful because it can simplify the grammar production rules considerably.

Production rules are specified using a BNF-like notation. An simple example of the notation is shown below:

```
StateExpr    statement -> expression_statement;  
StateSelect  statement -> selection_statement;  
StateIter    statement -> iteration_statement;
```

As you can see, each production is described by a single statement having an associated label (the left-most name). The right-hand side of each production describes the grammar to be recognized, whereas the left-hand side describes the reduction. In English, these lines mean that a *statement* can be an expression statement, a selection statement, or an iteration statement. Naturally, a complete grammar consists of dozens or hundreds of such production rules.

The parser generated by VP++ uses left-right parsing with an "infinite" look-ahead, known simply as LALR(k). This allow VP++ to parse many grammars considered ambiguous by `yacc`, which has only a single token look-ahead. You can select whether the generated parser interfaces to C, C++, Java, Delphi, or Visual Basic (via an ActiveX control). The Java interface - along with faster operation - are the primary features that were added to from version 2.0 to 2.1.

Unique to VP++ is its interactive environment, which provides an editor for creating or modifying a rule file. Once a rule file has been created can be compiled. The compiler checks for syntax errors, duplicate lexical tokens, and unreachable productions. Most importantly, it builds a parse tree and checks for shift-reduce and reduce-reduce conflicts that signify ambiguities in the grammar.

VP++ provides two ways for you to understand and eliminate grammar conflicts. The *Conflict View* window opens automatically if conflicts are detected during compilation. This window provides a list of conflict states and lets you activate the k-token look-ahead feature to attempt a resolution. For additional help with conflict resolution, the *Conflict Trace View* window can be opened. This view shows the entire parse tree that leads to a conflict, highlighting its cause.

The VP++ environment makes detailed debugging considerably easier than with `lex/yacc`. You can single-step through an input test file, during which each lexing and parsing action is highlighted.

Moreover, you can set one or more breakpoints which will halt input file parsing and let you examine the current parsing state. Actually these features transcend debugging, providing a great learning tool for people new to parsing concepts.

There is much more to Visual Parse++ than the basic features I've described. The grammar file supports macros and precedences. The product includes online and printed documentation, as well as sample grammar files and a simple tutorial. Although VP++ runs under the Windows, the Professional Edition includes source code that allows parsers to be built for UNIX, OS/2, Macintosh, and virtually any other environment.

Even though like and use this product, I must also point out some of its shortcomings.

To begin with, the online documentation is lackluster and the printed documentation is not very helpful to novice users. I was hard-pressed to use the product effectively until I read the "bird" book³, a very instructive guide to lex and yacc. Also helpful, though more advanced, is the classic "dragon" book⁴. Because its input file format and operation are quite different than lex/yacc, VP++ should provide a more detailed introduction to the subject of parsing.

Speaking of VP++ differences, the default precedence of production rules is exactly reversed from that of yacc. This is a nuisance for anyone (like me!) converting a yacc grammar file to the VP++ format.

Although it's been improved from version 2.0, the application is not always well-behaved. Compiling and tracing are sometimes difficult - or impossible - to abort. The program does not always respond to commands such as minimization and maximization. Moreover, the display windows sometimes refresh very slowly or not at all. On the other hand, I've never seen the program exit prematurely or cause a protection violation of any kind.

I also have a few complaints about the user interface. The *Compile* button is adjacent to the *Open Test File* button on the toolbar. Maybe I'm clumsy, but I can't tell you how many times I've accidentally re-compiled my grammar file instead of opening a test file. This can be especially frustrating if you have a large grammar file that takes a long time to compile.

Although single-stepping is a great feature, it becomes laborious when running long test files. I would like an additional control that parses the input file to the point of the next recognized token or next reduction, whichever comes first. I would also like menu items that allow the default shift-reduce and reduce-reduce conflict settings to be used instead of stopping and prompting every time a conflict is encountered.

Perhaps because it has no real competition, Visual Parse++ suffers from some curious bugs and limitations. Nevertheless, I've found it to be an innovative, useful and reliable tool. Version 3 (to be released early in 1998) is expected to have many enhancements. Among these are natural-language parsing and more views of the lexing and parsing machines. Also anticipated is the addition of ready-made parsing solutions for C++ and Java, and an ActiveX control for parsing HTML files.

Product Information

Visual Parse++ Version 2.1
Sandstone Technology
939 Coast Boulevard, Suite 4C
La Jolla CA 92037
Tel: 800-988-9023 / 619-454-9404
Fax: 619-454-9467
Email: wildd@sand-stone.com
Web: <http://www.sand-stone.com>

Pricing

Standard:	\$299.00
Professional:	\$399.00
Cross-Platform:	\$699.00
V2.0 Upgrade:	\$49.00

Notes and References

1. Fitzpatrick, J. "Applying the ABC Metric", *C++ Report*, 9(6), June 1996.
2. ABC calculator programs may be downloaded from <http://www.braveidea.com>.
3. Levine, J.R., Mason, T., and Brown, D. "lex & yacc", *O'Reilly & Associates, Inc.*, Sebastopol, CA, 1995.
4. Aho, A.V., Sethi, R., and Ullman, J.D., "Compilers: Principles, Techniques, and Tools", *Addison-Wesley*, 1988.