An Attributed ELL(1)-Parser Generator

Henk Alblas and Joos Schaap-Kruseman

University of Twente, Department of Computer Science,
P.O. Box 217, 7500 AE Enschede, The Netherlands.
e-mail: alblas@cs.utwente.nl

Abstract

TCGS is an acronym for Twente Compiler Generating System. It has been developed at the Compiler Methodology Group at the University of Twente. The system consists of a scanner generator, a combined parser and attribute-evaluator generator, a standard routine module, and an interpreter. The standard routine module contains type definitions, routines for the storage and retrieval of declaration information, routines that support attribute evaluation, and routines that support code generation for a hypothetical stack machine. The linker builds a complete compiler in VAX Pascal from the generated attributed parser and scanner, the standard routine module, and possibly, user defined modules. The interpreter, useful in compiler courses, is used to execute programs in target code. This paper is devoted to the parser and attribute-evaluator generator.

The generated parsers are L-attributed ELL(1) recursive-descent parsers. ELL(1) parsers are an extension of LL(1) parsers in the sense that they are derived from Extended Context-Free Grammars (abbreviated to ECFGs) instead of normal context-free grammars. ECFGs are also called Regular Right Part Grammars. An attributed ECFG is an ECFG augmented with attributes and attribute evaluation rules. An attributed ECFG can be used to specify both context-free and context-sensitive language properties, as well as the translation from source code to target code. The attribute grammar is restricted to be L-attributed, which means that the information flow, specified by the attributes, goes strictly from left to right.

Attribute evaluation rules are represented by actions. Actions must be defined in an action definition list. An action definition consists of an action name and a parameter list. Action parameters are specified as input or output parameters. An action consists of an action name and an argument list which corresponds with the parameter list of the action definition. The output arguments are attribute occurrences. The input arguments are attribute occurrences or constants. For every action there must be a corresponding Pascal procedure declaration in a definition file. The actions are included in the production rules. The position of an action determines when it is executed. Syntactically, an action can be viewed as another

nonterminal whose production rule has an empty right part. The parser generator verifies for each action whether all its input arguments are assigned. It has no provisions to verify whether the output arguments of the action procedures are assigned, because the procedure bodies exceed the field of view of the generator system.

Besides the usual inherited and synthesized attributes, TCGS also supports global and local attributes. Inherited and synthesized attribute occurrences can be assigned only once. An inherited attribute occurrence can be assigned in the production rule in which it occurs in the right part. A synthesized attribute occurrence can be assigned in the production in which it occurs in the left part. Global attributes can be referenced from every position in the grammar, and they may be assigned several times. Local attributes play a local role in a production rule, and they must be declared as such in the production rule. Local attributes are introduced to solve the problems caused by syntactic operators. These problems could also have been solved by using global attributes. The drawback of global attributes is, however, that they fail in the case of recursive grammars. Another solution could have been to let a special attribute of the left-side nonterminal play the role of a local attribute. This means, however, that such an attribute will be assigned several times. This is in conflict with the requirement that inherited and synthesized attributes are assigned only once.

The parser generator performs a reachability test and an availability test on the attributed input grammar. In the reachability test, the generator system verifies for every attribute occurrence which acts as a parameter of an action, whether reference to this attribute occurrence is allowed from this position in the production rule. In the availability test, the generator system verifies for every production rule whether all synthesized attribute occurrences of its left-hand side nonterminal and all inherited attribute occurrences of its right-hand side nonterminals are defined. Moreover, for every action it verifies whether the input arguments are available if the input text is parsed from left to right.

To implement the ELL(1)-test the TCGS parser generator builds a structure tree for every production of the input grammar and uses attributes which represent functions *empty*, *first* and *follow* to apply the ELL(1) test.

TCGS generates attributed parsers which support both syntactic and semantic error recovery. At any time during parsing the parser keeps a set of possible look-ahead symbols. If the next symbol that is read by the parser is not expected, but is in the look-ahead set, then the parser assumes that a symbol is missing, inserts this symbol and continues parsing. If, however, the not-expected symbol is not in the look-ahead set, then the parser assumes that the next symbol is superfluous, deletes it, and continues parsing. A TCGS parser continues attribute evaluation after encountering a syntactic error. This means that the parser keeps verifying the context-sensitive and semantic integrity constraints (expressed by attributes), even after program text has been deleted or inserted.

TCGS has been used both as a tool to construct compiler front ends, and in teaching compiler construction courses. Students have found TCGS to be easy to use. The system is still under development. Currently, facilities for the generation of abstract syntax trees are being added.