AGile, a structured editor, analyzer, metric evaluator, and transformer for Attribute Grammars

André Rocha, André Santos, Daniel Rocha, Hélder Silva, Jorge Mendes, José Freitas, Márcio Coelho, Miguel Regedor[2], Daniela da Cruz, and Pedro Rangel Henriques[1]

[1] University of Minho - Department of Computer Science,
Campus de Gualtar, 4715-057, Braga, Portugal
{danieladacruz,prh}@di.uminho.pt
[2] University of Minho - Department of Computer Science,
Master Course on Language and Grammar Engineering (1.st year)

Abstract. As edit, analyze, measure or transform attribute grammars by hand is an exhaustive task, it would be great if it could be automatized, specially for those who work in Language Engineering. However, currently there are no editors oriented to grammar development that cover all our needs.
In this paper we describe the architecture and the development stages of AGile, a structured editor, analyzer, metric calculator and transformer for attribute grammars. It is intended, with this tool, to fill the existing gap.
An AnTLR based attribute grammar syntax was used to define the input for this system. As soon as the user types the grammar, the input is parsed and kept in an intermediate structure in memory which holds the important information about the input grammar. This intermediate structure can be used to calculate all the metrics or to transform the input grammar.
This system can be a valorous tool for those who need to improve the performance or functionalities of their language processor, speeding up the difficult task of defining and managing a language. Features like highlighting, automatic indentation, on-the-fly error detection, etc., also adds efficiency.

1 Introduction

Editing an Attribute Grammar is a long and complex hard task. So it is important to provide basic support to make the editing easier. However, even more important is to assist the language engineering in designing and developing a language with quality. This requires that the grammar development environment (GDE) incorporates knowledge from grammar engineering, like analysis, visualization, metric evaluation, and transformation.

This article describes AGile, a text editor built to assist in the grammar writing process, comprising four main features as follows: *Syntax and lexical awareness*: syntax highlighting, automatic indentation, lexical integrity checking, syntactic integrity checking; *Metrics evaluation* for quality assessment (derivation Rules analysis: both size and structural parameters should be measured and metrics evaluated; attribute analysis: evaluate attribute-related metrics and other tasks — generate dependence graph, ...);

INForum 2010 - II Simpósio de Informática, Luís S. Barbosa, Miguel P. Correia (eds), 9-10 Setembro, 2010, pp. 197–200

Derivation Rules transformation: transformations upon types of rules, namely eliminating unitary rules, eliminate left/right recursion, ...; *Visualization*: dependence graphs, for both the syntactic and the semantics components, will be built and displayed.

AGile GDE is a project under development in the context of a master course on Language Engineering. The motivation for this work is the application of all the theoretical concepts on grammar engineering (quality assessment and metrics) and on software analysis and transformation (analyzers, internal representations, transformers and visualizers).

AGile parses the input grammar to check its syntactic and semantics compliance with the meta-language defined (in our case, we have adopted AnTLR [1] meta-language syntax); if errors are detected during this phase, descriptive messages will be generated. Other important AGile components are the metrics evaluator, the grammar transformation and the visualizer. To support all these operations, the system generates, during parsing, an attributed abstract syntax tree (AST) and creates an Identifier Table (IdTab)). The operations referred above will be executed on this intermediate representation.

The paper has, after this, 5 sections. Section 2 describes the syntax-directed editor. Section 3 discusses the metrics evaluated by AGile. Section 4 introduces the rule transformations implemented by the GDE proposed. Section 5 very briefly present the visualization provided at moment. As usual, Section 6 closes the paper.

2 Editor

The first component of AGile is a text editor. Editors are tools that give us the ability to collect, prepare and arrange material for storing objects in a computer to be processed afterwards.

In our project the objects we want to edit are structured documents instead of unstructured text; actually, we need to process structured texts representing grammars. So we decided to developed an editor that could help to create and maintain that structure.

Structure text editors can be classified into different categories [2–4]: structure-aware editors; syntax-directed editors; language-based editors.

Our editor falls on the second category and it treats grammar rules in an analytical way, meaning that we are using an intermediate representation to offer the user the functionality described bellow.

To help in the process of writing a grammar, our editor uses the knowledge about the meta-grammar to offer the following set of features: line numbering — useful for debugging, allowing a quick identification of a line; syntax highlighting — the editor sends the text to the analyzer producing an intermediate structure obtained from the abstract syntax tree, then this structure is traversed producing tokens colored differently according to their lexical/syntactic role; automatic indention of the text according to the hierarchical relationship between components.

3 Metrics Evaluator

Among other factors that affect the software quality, its complexity has a large influence, and thus metrics were designed to assess the software complexity.

In the context of grammar quality, Power and Malloy have defined in [5] a set of metrics, derived from the above mentioned software metrics. In this paper, we do not follow strictly Power and Malloy's approach. We took there classification as a basis, but we go further, defining a set of 10 metrics classified in 3 types: size, shape and lexicography metrics.

One of the main components of AGile is the evaluation of size metrics on a context-free grammar, edited under this grammar processing environment. The first set of metrics is related with the size of the grammar G, and the second one is concerned with the size the Parser derived from the grammar G.

The set of metrics metrics evaluated by AGile are the following: **#T** — Number of terminal symbols; **#NT** — Number of non-terminal symbols; **#P** — Number of productions; **#PU** — Number of unitary productions; **$RHS**$_{avg}$ — Average size of the right hand side; **$Alt**$_{avg}$ — Average number of alternatives for each symbol; **Fan-in & Fan-out** — Number of dependencies between symbols from the dependence graph; **#TabLL(1)** — Size of the LL(1) parse table; **#RD** — Size of the Recursive Descendant parser.

In this way the user obtains easily and in an interactive mode immediate feedback about his grammar. Thus, the user is aware of the options that is doing when constructing the grammar.

4 Transformer

When we are writing a grammar for a certain language, the one that we wrote is not always the best when taking into account factors like: grammar metrics, grammar comprehension, or even the efficiency of the future parser. In this context, it is important to consider some transformation techniques in order to make our grammar a better one.

In AGile were introduced two different types of transformations. the *rename of the name* of a Terminal, a Non-Terminal, or Attributes; the *elimination of Unitary Productions*. The first transformation can increase significantly the clarity of the grammar, making easier the comprehension and maintenance tasks – it increases the grammar quality. When we have a long grammar and we want to change the name of a symbol that is mentioned in a lot of productions, it would be a time consuming and exhausting task to change all of them by hand. Automatizing it, reduces effort and at the end assures a complete replacement. The second transformation produces a grammar more difficult to understand, but the advantage is that from the reduced grammar (with less Productions and Non-Terminals) we can generate a more efficient Processor.

AGile computes automatically the metrics for the new grammar, allowing the user to compare both version (the original and the transformed).

5 Visualizer

To complement the comprehension of the grammar under development, this AGile component produces and displays a visual representation of a grammar. It is well known from the literature and practice that the visualization of dependence graphs offers an effective help for program comprehension. Extending the same idea to grammars, our GDE also offers a functionality to show the Dependence Graphs between Symbols, marking in each node its values of Fan-in and Fan-out.

6 Conclusion

This paper introduced AGile, a structured editor, analyzer, metric calculator and transformer for attribute grammars. Mainly, this tool allows the user to write a grammar, based on its specific syntax, and apply several actions over it.

Features like syntactic-directed edition, metrics evaluation and form transformations make AGile a useful tool for studying and formal grammars.

According to [6], grammar metrics have been introduced to measure the quality and complexity of a given grammar in order to direct the grammar engineering. Computing metrics since the early stages of the development of a given grammar, allows to improve the grammar and to avoid some undesirable features that otherwise would only be perceived later on.

Dependence graphs, for syntactic dependencies between grammar symbols or semantic ones between attributes, can also be generated and displayed to simplify the understanding of the grammar edited.

A lot of work still needs to be done to realize all our ideas, however the theoretical foundations are established and the prototype is promising.

References

1. Parr, T.: The Definitive ANTLR Reference: Building Domain-Specific Languages. First edn. Pragmatic Programmers. Pragmatic Bookshelf (Maio 2007)
2. Donzeau-Gouge, V., Huet, G., Kahn, G., Lang, B.: Programming environments based on structured editors: the mentor experience. Rapport de Recherche 26, INRIA, Rocquencourt (July 1980)
3. Teitelbaum, T., Reps, T.: The cornell program synthesizer: A syntax-directed programming environment. Communications of the ACM **24**(9) (Setembro 1981)
4. Reps, T.: Generating Language-Based Environments. PhD thesis, Cornell University (1982)
5. Power, J.F., Malloy, B.A.: A metrics suite for grammar-based software: Research articles. J. Softw. Maint. Evol. **16**(6) (2004) 405–426
6. Cervelle, J., Crepinsek, M., Forax, R., Kosar, T., Mernik, M., Roussel, G.: On defining quality based grammar metrics. In: Proceedings of the 2nd Workshop on Advances in Programming Languages (WAPL'2009), Mragowo, Poland, IEEE Computer Society Press (October 2009) 651–658