Graphical UML View from Extended Backus-Naur Form grammars

Fathi Essalmi & Leila JEMNI BEN AYED
Fathi.essalmi@isg.rnu.tn Leila.jemni@fsegt.rnu.tn
Research Unit of Technologies of Information and Communication (UTIC)[1]
Ecole Supérieure des Sciences et Techniques de Tunis , University of TUNIS
5, Av. Taha Hussein, B.P. 56, Bab Menara, 1008 Tunis, TUNISIA
fax: +216 71 39 25 91 phone: +216 71 49 60 66

Abstract

This paper addresses the graphical representation of Context-Free Grammars (CFG) written in extended Backus-Naur Form (EBNF), using UML class diagrams. These diagrams can help understand the grammar for users who are not familiar with concepts of EBNF grammars. The additional structure afforded by UML could help to clarify how the derived language is constructed and allows learner to build significance relations between the concepts presented. We propose derivation rules of EBNF grammars into UML class diagrams which will be illustrated over an example of a grammar written in EBNF.

1. Introduction

The Extended Backus-Naur form (EBNF) [3] is nowadays the most rigorous way to define syntax of programming languages. Hopefully, techniques that are based on graphical notations are often perceived as more intuitive and readable than formal notations. In [4], A. van Lamsweerde states the popularity of techniques based on tabular formats and diagrammatic notations. Similarly, in [10], Zimmerman and his co-authors have conducted an experimental evaluation that confirms the advantages of tabular and diagrammatic notations. This has been confirmed by several cognitive science studies [7]. In [9], Yong Xia and Martin Glinz have proposed a mapping from graphical languages to EBNF aiming at the elimination of inconsistencies and ambiguities in UML diagrams. One problem with this approach is that such translations, aiming to be as comprehensive as possible, tend to result in unnatural that are hard to understand and reason about. Although perhaps less natural at first sight, we believe that it is also interesting to derive UML models from EBNF grammars. For example, the additional structure afforded by UML could help to clarify how the derived language is constructed. In this paper, we investigate the reverse approach; going from an EBNF grammars to a graphical representation in UML [8] class diagrams. We propose a transformation approach which allows the generation of UML class diagrams from grammars written in EBNF. The use of UML (Unified Modeling Language) to present knowleges enables us to profit from MOF (Meta Object Facility) and graphical presentation in UML class diagrams allows learner to built significance relations between the concepts presented [5]. In the class diagram, the learners visualize graphically different relations between terminals and non-terminals. Among these relations: the composition, and the generalization. Our approach results a class diagram that is a connected graph, for that reason, the learners can visualize also the relations between a non-terminal and other non-terminals and terminals that dos not appear in its definition. The visualization of the different relations allows the learner to understand the CFG and the generated programming language.

For instance, this approach is very useful for a professor to present concepts of some languages which can be derived from BNF (Backus Naur Form) or EBNF form grammars such as the B language or any modeling or programming language. The paper is structured as follows: section 2 presents derivation rules of grammars written in BNF or EBNF form in to UML class diagram. In section 3, we illustrate our approach through an example.

2. Translation Rules of BNF and EBNF Grammars to UML class diagram

In this section, we present rules that generate a class diagram from a context-free grammar written in EBNF. A context-free grammar [6] is a 4-tuple G = (V, Σ, R, S), where:

- V: A finite set of non-terminals.
- Σ: A finite set of terminals.
- R: A finite set of production rules.
- S: A start symbol.

The Backus-Naur Form (BNF) is used for the definition of context-free grammars. The Extended Backus-Naur Form (EBNF) adds the regular expression syntax [3]. The BNF contains the following notation:

- ::= : Definition of non-terminals.

[1] www.esstt.rnu.tn/utic

Proceedings of the Sixth International Conference on Advanced Learning Technologies (ICALT'06)
0-7695-2632-2/06 $20.00 © 2006 IEEE

- Space ( ) : Concatenation of two grammatical units (terminals or non-terminal)
- | : Choice between two definition of a non-terminal

The EBNF adds the following notations:

- * : n concatenations of grammatical units, $n \in [0..+\infty]$
- + : n concatenations of grammatical units, $n \in [1..+\infty]$
- [ ] : n concatenations of grammatical units, $n \in [0..1]$

The generation of UML class diagram from BNF or EBNF form uses following rules.

Rule 1. ***Generation of classes***

- *Condition 1:* A non-terminal NT $\in$ V will be represented in UML by a class called NT in the class diagram.
- *Condition 2:* An expression E $\in (V \cup \sum)^+$ that verifies |E| >1, and E is an image of a non-terminal NT in R, and NT have more than one image, will be represented by a class called E in the class diagram.

Rule 2. ***Generation of Attributes***

- *Condition 1:* If a production rule is of the form NT::=$a_1\ a_2\ ..a_n$ where $a_1\ a_2\ ..a_n \in \sum$, then the terminals $a_1\ a_2\ ..a_n$ will be represented by attributes of the class NT.
- *Condition 2:* If a production rule is of the form NT::= $a_1 \mid a_2 \mid..a_n \mid\ldots$ where $a_1\ a_2\ ..a_n \in \sum$, then the terminals $a_1\ a_2\ ..a_n$ will be represented by attributes of the class NT, and the class NT will have the stereotype <<enumeration>>.
- *Condition 3:* If an expression E is represented in the class diagram by a class, then each terminal that appears in E will be represented by an attribute of the class E.

Rule 3. ***Generation of relations***

- *Condition 1:* If a production rule is of the form NT::= $E_1 \mid E_2 \mid... E_n \mid\ldots$ where $E_1, E_2... E_n \in (V \cup \sum)^*$ and $E_1, E_2... E_n$ are represented by classes in the class diagram, then the relations between NT and $E_1, E_2... E_n$ will be: generalisation(NT, E_1), generalisation(NT, E_2), ... generalisation(NT, E_n)
- *Condition 2:* If a production rule is of the form NT::= $NT_1\ NT_2\ ...NT_n$ where $NT_1, NT_2,...NT_n \in V$, then the relations between NT and $NT_1, NT_2,...NT_n$ will be: Composition(NT, NT_1), Composition(NT, NT_2), ... Composition(NT, NT_n)
- *Condition 3:* If an expression E is represented in the class diagram by a class, then the relation between E and each non-terminal NT that appears in E will be the relation: Composition (E, NT).

Rule 4. ***Generation of multiplicity***

- *Condition 1:* If t $\in (V \cup \sum)^*$ appears in the definition of NT of the form E*, the multiplicity of E will be * if E is an attribute, and the multiplicity of the relation between E and NT beside E will be * if E is a class.
- *Condition 2:* If t $\in (V \cup \sum)^*$ appears in the definition of NT of the form E^+, then the multiplicity of E will be 1..* if E is an attribute, and the multiplicity of the relation between E and NT beside E will be 1..* if E is a class.
- *Condition 3:* t $\in (V \cup \sum)^*$ appears in the definition of NT in the form [E], the multiplicity of E will be 0..1 if E is an attribute, and the multiplicity of the relation between E and NT beside E will be 0..1 if E is a class.

The generation process generates the classes from some features of a grammar by applying some rules (each non terminal which appears in left part of a production rule is translated in to a class). Then uses this diagram in order to evolve it and treats each production rule to generate sub classes, attributes, associations, states and other relations (composition, dependence,...).

3. Example

In this section, we use some production rules extracted from the B grammar [1], and we apply the transformation rules presented in section 2. We build a graphical representation of concepts in the grammar and their relations using UML class diagram.

Composant ::= Machine_abstraite | Raffinement | Implantation

Machine_abstraite ::= "MACHINE" En-tête Clause_machine_abstraite* "END"

Clause_machine_abstraite ::= Clause_constraints | Clause_sees | Clause_includes | Clause_promotes | Clause_extends | Clause_uses | Clause_sets | Clause_concrete_constants | Clause_abstract_constants | Clause_properties | Clause_concrete_variables | Clause_abstract_variables | Clause_invariant | Clause_assertions | Clause_initialisation | Clause_operations

The application of the transformation rules on the first production, results a first level graphical representation in UML class diagram presented in Figure 1.

Then we evolve diagram 1 by applying the transformation EBNF-UML on the second production rule. This results on the model given in figure 2.

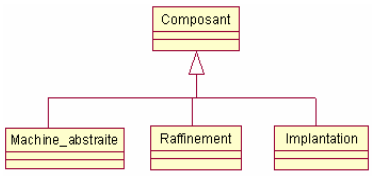

Figure 1. Class diagram 1

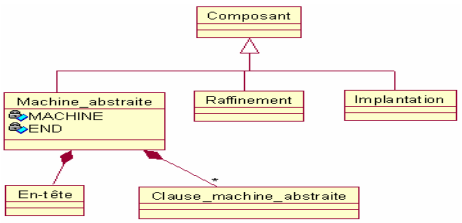

Figure 2. Class diagram 2

By applying the transformation EBNF-UML on the third production rule we obtain the model in figure 3.

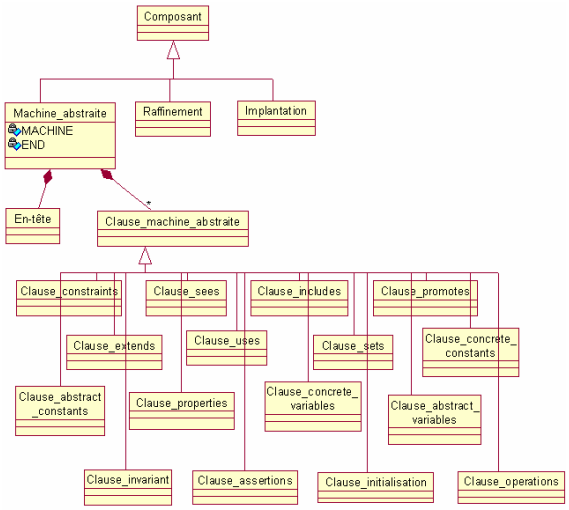

Figure 3. Class diagram 3

In the Figure 3, we visualize in the same diagram the three production rules. Additional knowledge are presented such that the relation between the non-terminal Machine_abstraite and the non-terminal Clause_operations. The operations clause is a particular abstract machine clause that can be used in an abstract machine.

4. Conclusion

Although grammars written in EBNF provide excellent techniques for the precise description of languages, understanding these descriptions is often restricted to users familiar with such notations. This paper presented a technique that helps building a graphical representation of concepts in the grammar and their relations using UML class diagram. This diagram is expected to be more intuitive and readable than the original formal notation. It is intended as an accompanying documentation in a process which involves customers not trained in EBNF or BNF grammar forms and then helps them to generate languages. It can also help teachers of modeling and programming languages to elaborate concepts their relations and to explain them using graphical representation. This paper has proposed transformation rules of BNF and EBNF grammars to UML class diagram. The automation of the transformation process is a perspective of this work. Actually a prototype is developed.

5. References

[1] CLEARSY, *Manuel de référence du langage B*, Version 1.8.5, 2003

[2] H. FEKIH, L. JEMNI BEN AYED and S. MERZ, "Transformation of B specifications into UML class diagrams and State Machines", *21 st Annual ACM Symposium on Applied Computing*, 23-27 Avril 2006, Dijon, France.

[3] ISO/IEC, *EXTENDED BNF 1996* www.dataip.co.uk/Reference/EBNF.php

[4] A.V. Lamsweerde, "Formal Specification: a Roadmap", *ICSE'00: The Future of Software Engineering Track*, ACM Press, pp. 147-159, 2000.

[5] D. Legros and J. Crinon, *Psychologie des apprentissages et multimdia*, Armand Colin /VUEF, Paris, 2002, ISBN 2-200-26248-5

[6] C.-H. L. Ong, *Context-free grammars*. http://users.comlab.ox.ac.uk/luke.ong/teaching/moc/cfg2up.pdf

[7] M. Petre. "Why looking isn't always seeing: Readership skills and graphical programming", *Communications of the ACM*, Vol 38 N° 6, pp. 33-44, 1995.

[8] G. Rady Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language user guide,* Addison Wesley Longman Publishing Co., Inc., 1999.

[9] Y. Xia and M. Glinz, "Rigorous EBNF-based Definition for a graphic Modeling Language", *Proceedings of the Tenth Asia-Pacific Software Engineering Conference (APSEC'03), IEEE*, 2003.

[10] M. K. Zimmerman, Kristina Lundqvist, and Nancy G. Leveson. "Investigating the readability of state-based formal requirements specification languages", *ICSE'02: 22rd International Conference on Software Engineering*, pp 33- 43. ACM Press, 2002.