Graphical UML View from Extended Backus-Naur Form grammars

Fathi Essalmi
Fathi.essalmi@isg.rnu.tn
Research Unit of Technologies of Information and Communication (UTIC)
Ecole Supérieure des Sciences et Techniques de Tunis, University of TUNIS
5, avenue Taha Hussein, B.P. 56 Bab Menara 1008 – Tunis - Tunisia

Leila Jemni Ben Ayed
Leila.jemni@fsegt.rnu.tn

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 built 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) [6] 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 [1], A. van Lamsweerde states the popularity of techniques based on tabular formats and diagrammatic notations. Similarly, in [7], 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 [8]. In [9], Yong Xia and Martin Glinz have proposed a mapping from graphical language 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 [5] 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 [4]. The additional structure afforded by UML could help to clarify languages derivation.

For instance, this approach is very useful for a professor to present concepts of some languages which can be derived from BNF or EBNF form grammars such as the B language or any modeling or programming language. This 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 for BNF and EBNF grammars

In this section, we present rules that generate a class diagram from a context-free grammar written in EBNF. A context-free grammar [2] is a 4-tuple \( G = (V, \Sigma, R, S) \), where:

- \( V \): A finite set of non-terminals.
- \( \Sigma \): A finite set of terminals.
- \( R \): A finite set of production rules.
- \( S \): A start symbol.

The BNF (Backus-Naur Form) is used for the definition of context-free grammars. The Extended BNF (EBNF) adds the regular expression syntax [6]. The BNF contains the following notation:

- \( ::= \): Definition of a non-terminals.
- \( \{ \} \): Choice between two definition of a non-terminal.
- \( \{ \cdot \} \): Choice between two definition of a non-terminal.

The EBNF adds the following notations:

- \( * \): \( n \) repetition (or concatenation) of grammatical unit, \( n \in [0..+\infty] \)
- \( + \): \( n \) repetition (or concatenation) of grammatical unit, \( n \in [1..+\infty] \)
- \( [ \cdot ] \): \( n \) repetition (or concatenation) of grammatical unit, \( n \in [0..1] \)
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 \( E \in (V \cup \Sigma)^* \) 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 in the form of \( NT::=a_1 \ a_2 \ldots a_n \) where \( a_1, a_2 \ldots a_n \in \Sigma \) the terminals \( a_1, a_2 \ldots a_n \) will be represented by attributes of the class \( NT \).
- Condition 2: If a production rule in the form of \( NT::=a_1 | a_2 | \ldots a_n | \ldots \) where \( a_1, a_2 \ldots a_n \in \Sigma \) the terminal \( a_1, a_2 \ldots a_n \) will be represented by attributes of the class \( NT \), and the class \( NT \) will have the stereotype \(<\text{enumeration}>\).
- Condition 3: If an expression \( E \) is represented in the class diagram by a class, 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 \( NT::=E_1 \ E_2 \ldots E_n \ldots \) where \( E_1, E_2\ldots E_n \in (V \cup \Sigma)^* \) and \( E_1, E_2\ldots E_n \) are represented by classes in the class diagram, then the relations between \( NT \) and \( E_1, E_2\ldots E_n \) will be:
  - generalisation(\( NT, E_1 \)),
  - generalisation(\( NT, E_2 \)), ...
  - generalisation(\( NT, E_n \))
- Condition 2: If a production rule is of the form \( NT::=NT_1 \ NT_2 \ldots NT_n \ldots \) where \( NT_1, NT_2\ldots NT_n \in V \) and the class \( NT \) will have the stereotype \(<\text{enumeration}>\), then the relations between \( NT \) and \( NT_1, NT_2\ldots NT_n \) 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 \( t \in (V \cup \Sigma)^* \) 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 \( t \in (V \cup \Sigma)^* \) 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.

3. Example
In this section, we use some production rules extracted from the B grammar [3], and we apply the transformation process presented in section 3. We build a graphical representation of concepts in the grammar and their relations using UML class diagram.

Production rule 1
Composant ::= Machine_abstraite | Raffinement | Implantation

Production rule 2
Machine_abstraite ::= “MACHINE” En-tête Clause_machine_abstraite* “END”

Production rule 3
Clause_machine_abstraite ::= Clause_constraints | Clause_sees | Clause_includes | Clause_promotes | Clause_extends | Clause_uses | Clause_sets | Clause_concrete_variables | Clause_abstract_variables | Clause_properties | Clause_concrete_constants | Clause_abstract_constants | Clause_initialisation | Clause_operations

The application of the approach presented in section 3, results a first level graphical representation in UML class diagram as follows:

Figure 4. Class diagram 1

Then we evolve diagram1 by applying the transformation EBNF-UML on the second production rule. This results on the model given in figure 5.

Figure 4. Class diagram 1
By applying the transformation EBNF-UML on the third production rule we obtain the model in figure 6.

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 a transformation rules from EBNF or BNF notations to UML class diagram and a process giving the generation steps. The automation of our approach is a perspective of this work. Actually a prototype is developed.

5. References