CONTROL LANGUAGE FOR PRIMITIVES AND GRAPHICAL WINDOWS

Vasile GURDUZA, Patricia BARACU, Nicoleta TIRDEA, Vasilica MUNTEANU

Technical University of Moldova

Abstact: *In this paper is investigating the design of a specific language which control primitives and graphical windows that will allow simulation of it. The language is described by semantic rules, grammar, lexer using ANTLR, all of this is a key of developing a specific language.*

Keywords: *Antlr, primitive, shape, window, finite automata, grammar.*

Introduction

Nowadays, in our society is growing up the necessity of new languages describing a specific domain, so now all of us are focused on creating and developing new Domain Specific Languages.

Generally speaking, DSL represent a computer language, that is specificed on certain domain, being able to solve a specific problems. In this case, one of the big problem today,it is to control different types of primitives like point, line to complicated one triangles, squares [1].

Hence, the importance of domain specific languages is so great, because of it we thought to improve this domain by solving a problem in certain area, the DSL was named Control language for primitives and graphical windows. The meaning of this DSL is to work with primitives and to generate different graphical windows with one or more primitives in it. This is intended for people who working daily with kind of programming languages like specialists or a simple one who just do it like hobby, improving their knowledge and aptitudes.

Language for control primitives and graphical windows is characterized by set of rules that emphasize its individuality as a programming language. Consequently, all variable that appear in the program must be declared at the top of program before its used. Its name must be different from names of methods or other words which was already declared and names always starts with a letter.

The most significant semantic rules are:

- Identifiers must be declared only once in the same purpose.
- All the object that should be drawn must be call before the method draw.
- Each method must be assigned to a window.

Additionally, from semantic perspective all programs must contain and start with a single keyword “program” followed by a name of this program starting with a letter. After that are declared only variables that are present in the program, next line contains all necessary statements (method calls and assignments). The last line of the program includes keyword “run” that means the end of the program and the subsequent start of the compiling process. Furthermore, all lines of code must begin from a new row, otherwise the compiling process will end with an error.

1. Grammar

A grammar of a domain specific language is

```
alpha_num
    : ALPHA
    | DIGIT
    ;

ALPHA
    : [a-zA-Z ]
    ;

DIGIT
    : [0-9]
    ;
```

Fig.1. The apha_num grammar.

Grammar denotes syntactical rules that need to be implemented in order do have a functional DSL. All grammar was implemented using mostly ANTLR tool [2]. It should be noted that above is represented just a small part of the grammar, apha_num grammar, also below is shown Finita Automata for this part of grammar.

Lexical alpha may contain elements from the uppercase or lowercase letters, and alphaNum can be an alpha- or digits and the syntax must end with the ‘ ; “ sign.

Lexical analyzer is intended to analyze and dissolve the code of the program character by character that subsequent are known as lexemes (main keywords that this program does contain).

Finite Automata a special machine responsible for a finite number of states [1]. Finite Automata can be of two types NFA (Nondeterministic Finite Automata) and DFA (Deterministic Finite Automata). We worked with a NFA because it allows to transit from one state to another through certain inputs, also it give the possibility to transit without reading symbols through null (or ε) inputs.

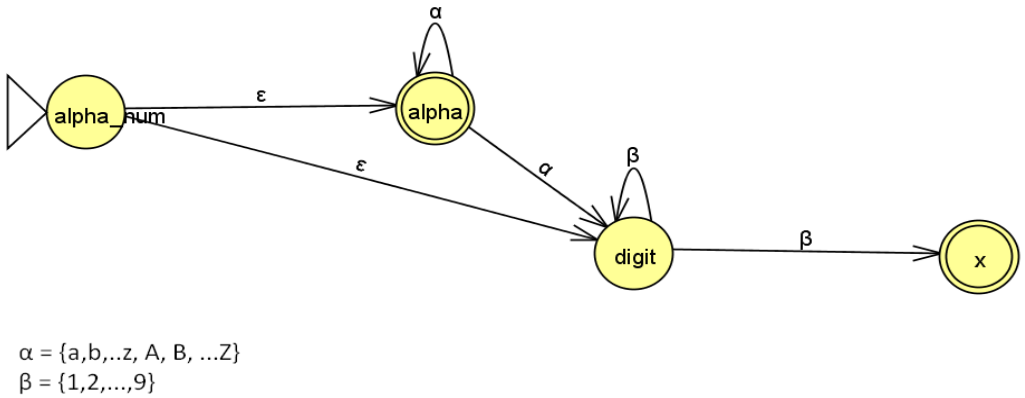

Fig.2. Finite Automata for alpha_num.

In figure 2 alpha_num is an initial state it can go to digit state or alpha, which is a final state. It can transit with α (choosing one or more form a to z, also uppercase ones) input to next state (digit), then with β (choosing one or more digits from 1 till 9) input getting the final state.

2. ANTLR

The reference grammar for the DSL was defined in ANTLR [3]. This is a tool that takes as input a context-free grammar that specifies a language and generates as output source code for a recognizer of that language. ANTLR can generate lexers, parsers and tree parsers [3].

ANTLR reads the grammar and generates a program that reads an input stream and returns an error if the input stream does not conform to the syntax specified by the grammar. It is used to generate the parse tree for the language. Once there is an abstract syntax tree (example from the figure 3 below), it is possible to work on it and add modifications in order to ensure the original code was correct; to interpret it or to compile it. A further step is to write actions for the grammar in a specific programming language, in order to emit instructions in a specific language.

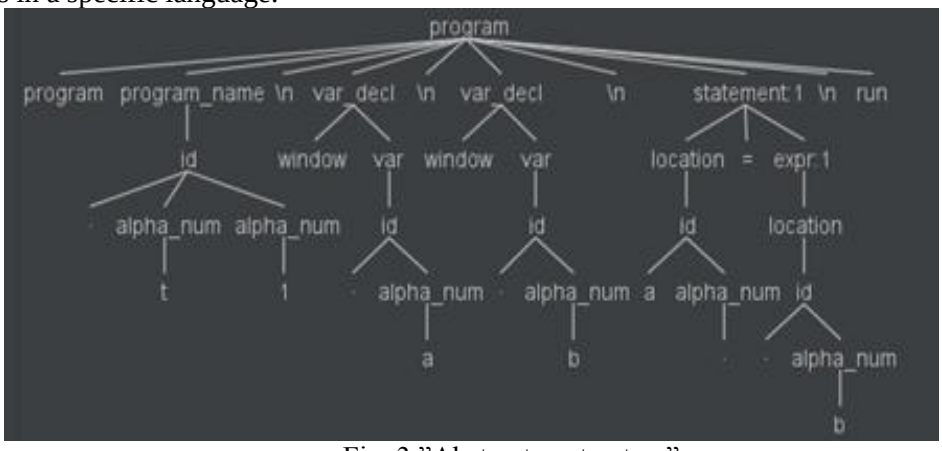

Fig. 3 ”Abstract syntax tree”.

The grammar for the project is divided in two parts: the lexer and the parser. The lexer defines the lexical analysis of the code by separating a stream of characters into different tokens. The parser defines rules of grammar for these tokens and determines whether these statements are semantically correct.

By using ANTLR in the process of developing the DSL it is easier to understand how the code is recognized and properly understood by an interpreter.

Conclusions:

Working on Domain Specific Languages is about solving a real-specific problem from nowadays. The most important things, which every DSL needs in order to be created, even more, the international programming languages like Java, C# and others has many requests in order to achieve their level, this is why the most known programming language has a lot of common things. For developing a new DSL is necessary to follow some rules: to have a grammar, semantic, parse rules and others. The purpose of this DSL is to work and control primitives and graphical windows. This language is generally defined by grammar written in Antlr, semantic rules and lexical analysed through finite automata. Therefore, this DSL is a key to solve problems to control graphical window and primitives for all who just learn this domain.

References

1. P. Linz ,"An Introduction to Formal Languages and Automata" Fifth Edition.
2. J.E.Hopcroft , R.Motwani, J.D.Ullman "Introduction to Automata Theory, Languages, and Computation " Second Edition.
3. T. Parr " The Definitive ANTLR 4 Reference".
4. A. M. Pitts "Regular Languages and Finite Automata".
5. D. C. Kozen " Automata and Computability".
6. J.Carroll, D. Long " Theory of Finite Automata With an Introduction to Formal Languages".
7. T.Parr "Language Implementation Patterns: Create Your Own Domain-Specific and General Programming Languages".
8. https://tomassetti.me/getting-started-antlr-cpp/.
9. https://en.wikipedia.org/wiki/ANTLR.
10. https://www.antlr.org/.