Structure Editor Generator aspect of ALISE

3K-7

Yasushi Kambayshi

Mitsubishi Research Institute

Introduction

A Language Independent Structure Editor(ALISE) is an interactive programming environment which supports the generation of structure editors and development of programs by using such structure editors.

In this paper, we present the structure editor generator aspect of ALISE. The editor generator in ALISE allows a user to define an editor for his/her own language. The editing procedure for a grammar is done using the graphic feature available in the Smalltalk programming environment.

The Language Description

In order to instantiate an editor, a description of the grammar for the language to be edited must be created. This is done through constructing a structured template on a display screen by choosing basic elements such as parallel elements, refinement elements, and serial elements, and typing key words.

The description of the language is also structured information; it defines the structure of the grammar of the target language in the same way the grammar expressed by the description defines the structure of programs in the target language. The description of the language is described by the notation, which contains the following basic elements: parallel elements, refinement elements, serial elements, and lexical elements. Therefore the structure editor generator is itself a structure editor, one which uses basic elements to edit the grammer for the target language.

The description of the language's syntax consists logically of two levels of structure. One level provides a description list, that is a list of the classes of elements that can replace placeholders. The classes represent the set of the legal operators and the set of templates that can replace placeholders in templates. The other level provides a description of the structure of each template. In this editor generator, both levels are described in a unified form. The basic description form is as Figure 1.

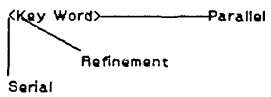

Figure 1: The basic description

The key words correspond to the left hand side of production rule in a context free grammar. The user specifies the name of the template and the name of class descriptions. This name used as the language command name and in the menu for programming phase.

Parallel structures represent both inherent parallelism and flow of control structures such as the case structure or the if-then-else structure. Parallel structures are also used for set notation. In other words, parallel structure express any disjunctive structures of a program.

Refinement structures support the creation of a program by step-wise refinement. By using this construct, a single statement can be recursively split into a sequence of statements. One advantage of introducing this construct is that a context-free grammar for the target language can be defined very clealy.

Serial structures are merely used to point to subsequent templates or expressions. In other words, serial structures express conjunctive structures of a program.

To edit the language description, ALISE provides a browser called ALISE browser which is similar to the Smalltalk standard system browser. The user uses a pointing device and a pop-up menu to edit the language description in this browser.

The Class Description

The classes are descriptions of legal operators which users use to fill in the placeholders in templates. The structure editor uses this class information when the user replaces a placeholder. It thereby ensures the syntactic correctness of the user's replacement.

Non-terminals are specified by using angle brackets as shown in Figure 2.

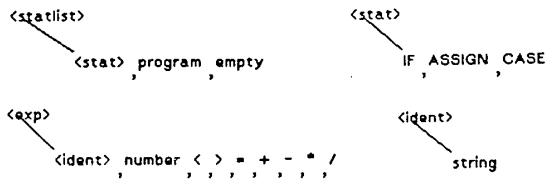

Figure 2: The class description

There is a set of primitive base types. They need not be defined to be used in a language description. These are "empty", "string" and "program". When using these base types, a user does not have to define regular expressions every time he/she defines a language description.

Structure of the Language

The main task for this editor generator is the construction of the structure for a language. This is done through building the necessary templates for the language. The user constructs the template he/she desires from the basic components, the parallel element, the refinement element, and the serial element. The meta-template is the basic description form which a description of the grammar is built, using the classes just defined.

On the ALISE browser, the user is required to select "key word" from the pop-up menu and to specify the name of it first. This specified key word becomes to be a template name then to be stored in ALISE environment. Then the user chooses one of the basic elements at a time to construct a template. When the user selects "key word", the cursor appears at the current node and waits for some text to be typed. When the user selects "parallel", "refinement", or "serial", a horizontal line, oblique line, or vertical line appears respectively on the workspace. To define a statement which has an optional field, such as the CASE statement, the user has to define two templates, one is for the mandatory part and other is for the optional part. Also one class description is required to ensure legal replacement. The Figure 3 shows an example of CASE template. Placeholders are indicated with angle brackets.

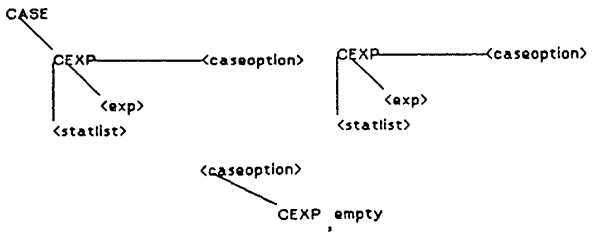

Figure 3: CASE template

Conclusion

The basic elements of the editor generator in ALISE is the essential elements, and are as powerful and flexible as BNF for the grammar description. ALISE's is much easier to understand than BNF because of its graphical representation.

Reference

Notkin, D. The Gandalf Project. *Journal of Systems and Software* 5(2), May 1985