Improvement of Decision Table Automatic Generation Tool VDTable for let in Statement

Yinuo Huang*, Tetoru Katayama*, Yoshihiro Kita†, Hisaaki Yamaba*, and Naonobu Okazaki*

University of Miyazaki, 1-1 Gakuen-kibanadai nishi, Miyazaki, 889-2192 Japan

†Tokyo Institute of Technology, 1404-1 katakura, Hachioji, 192-0914 Japan

E-mail: kouitsudaku@earth.cs.miyazaki-u.ac.jp, kat@cs.miyazaki-u.ac.jp,
kitayshr@stf.teu.ac.jp, yamaba@cs.miyazaki-u.ac.jp, oka@cs.miyazaki-u.ac.jp

Abstract

This research has expanded the applicable range of VDTable. VDTable is a tool of decision table automatic generation tool using VDM++ specification which is the lightweight formal methods VDM. We have improved VDTable to be able to correspond to let in statement, which is one of the VDM++ syntax, that the existing VDTable doesn’t support. We applied an application example to improved VDTable. From the result of the application example, the applicable range of VDTable has been expanded.

Keywords: VDM++, automatic generation, decision table, formal specification, VDTable, let in statement.

1. Introduction

Formal methods¹, in software development, is one of the means to describe the specification strictly. On the other hand, the decision table² is one of the testing techniques to exhaustively represent the logical combination included in software. But, for generating a decision table, it is necessary to understand the description content of the target system or specification. In extracting conditions and actions manually from specifications, it takes time and errors are prone to occur.

Therefore, our laboratory developed a VDTable³ (VDM Decision Table) to reduce this labor and time. VDTable can automatically generate the decision table, from the specification written in specification description language VDM++ (VDM++ specification). But, the VDTable, since many VDM++ syntax is not corresponded, its usefulness is limited.

To improve the usefulness, this paper expands VDTable. Specifically, VDTable is improved to be able to correspond to let in statement, which is one of the VDM++ syntax, that the existing VDTable does not support.

2. Introduction of existing VDTable

The flow of processing of VDTable is shown in Fig.1. VDTable consists of three parts: Parser, Converter, and DT-Generator.

Parser uses VDMJ⁴ to parse the VDM++ specification and outputs a parsing data. The parsing data has an abstract syntax tree. The abstract syntax tree holds analysis information such as the definition block type, argument, definition name and expression for each class definition. The parsing data to be used as input in the Converter.

Converter converts the parsing data outputted by Parser into internal expression data for analysis in order to facilitate extraction of conditions and actions and creation of truth values, which are necessary in generating a decision table. The internal expression data

© The 2017 International Conference on Artificial Life and Robotics (ICAROB 2017), Jan. 19-22, Sagaia Convention Center, Miyazaki, Japan

P - 271
for analysis is data obtained by dividing the parsing data in module units and expressing the expression of the definition main for each token as the minimum unit of the syntax.

DT-Generator defines extraction rules of conditions and actions. Firstly, it extracts conditions and actions, when its matching the condition extraction pattern or actions extraction pattern from the analysis internal expression data, and extracts String type array (condition array and action array), respectively. Next, DT-Generator stores conditions and actions in an array of String type. DT-Generator makes CA-Table (Condition Action Table), when DT-Generator extracts conditions and actions. CA-Table is a table which is correspondence of conditions and actions. CA-Table is three columns of condition index, token and action index. And then, DT-Generator generates truth-values based on this CA-Table. Finally, DT-Generator generates a decision table from the created condition and action, and the truth-values of the condition and action.

3. Improvement of VDTable

To correspond to let in statement, we extend Converter and DT-Generator. Specifically, we conduct the following three items.

- Proposal of data conversion rules for converting parsing data of let in statement to internal expression data for analysis
- Proposal of condition extraction rules and action extraction rules of let in statement

<table>
<thead>
<tr>
<th>Condition extraction pattern</th>
<th>Action extraction pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>Let condition 1 in condition 2</td>
<td>Name action</td>
</tr>
</tbody>
</table>

Table 1. Condition extraction rules and action extraction rules of let in statement.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 2. Example of creating a String type two-dimensional array that stores truth-values

- Proposal of truth table generation rules of let in statement

3.1. Proposal of data conversion rules for converting parsing data of let in statement to internal expression data for analysis

To parse let in statement, we use the Parser of the existing VDTable as it is. In VDTable, Converter converts parsing data of let in statement into internal expression data for analysis. In order to realize this, we propose data conversion rules of let in statement.

- In the data conversion of let in statement, when matching the token "let" from the parsing data, three items of "from immediately after 'let' to before 'in'”, "in", "immediately after 'in' to the last" are extracted.
- The part of name is extracted as one item.

These are saved as String type as internal expression data for analysis of let in statement.

3.2. Proposal of condition extraction rules and action extraction rules of let in statement

We propose condition extraction rules and action extraction rules of let in statement. Table 1 shows the condition extraction rules and action extraction rules of let in statement proposed in this paper. In the internal expression data for analysis, let the line above "in" be the condition 1, the line next to "in" be the condition 2, and let it be the condition extraction pattern. Also, the part after "name" is taken as the action extraction pattern.

© The 2017 International Conference on Artificial Life and Robotics (ICAROB 2017), Jan. 19-22, Sagaia Convention Center, Miyazaki, Japan
Next, DT-Generator extracts each condition according to the condition extraction rules. Then, DT-Generator extract actions according to the action extraction rules.

Finally, DT-Generator store the values of the truth value. In the proposed the truth table generation rules, it is assumed that when let in statement is described as "let condition 1 in condition 2", "if condition 1 and condition 2 are true (stores" Y"), the action is established (stores" X")," when Condition 1 is true (stores" Y"), condition 2 is false (stores" N"), action is not established (stores" -"). The reason is the movement of let in statement in the VDM specification is such that the condition 1 part is assigned to the condition 2. So, for condition 2, the condition 1 is always true. Therefore, when generating the truth table of let in statement, it is assumed that the condition 1 is always true, and when the condition 2 is true or false, the operation is determined or established.

The DT-Generator generates a decision table based on the extracted conditions and actions, and the truth values of the generated conditions and actions.

4. Application Example

The specification used for an application example is the specification described using VDM++ for triangle shape determination, and it is determine whether triangle by using let in statement. The specifications of shape judgment of the triangle is shown in Fig. 2. The result of parsing this specification with Parser is shown in Fig. 3.

Fig. 4 shows the result of converting the content of let in statement in the specification from the syntax analysis data in Fig. 3 to the analysis internal expression data. Condition 1 and condition 2, and contents of let in statement including action are saved as String type. It is understood that let in statement is correctly converted to the internal expression data for analysis.

Fig. 5 shows the result of applying the VDM++ specification of the application example to the improved VDTable. From the decision table of Fig. 5, it can be seen from Fig. 4 that the line before the token "in" is correctly extracted as condition 1 and the line following "in" as condition 2. Also, from Fig. 5, it can be seen that the portion after "name" is correctly extracted as an operation.

Finally, from the truth table of the decision table of Fig. 5, when the decision table shows the condition 1 and the condition 2 are true (stores" Y"), the action is established
6. Conclusion

In this research, VDTTable has been improved for the purpose of expanding the application range of VDTTable, an automatic generation tool for decision tables using the VDM++ specification. Specifically, we propose data conversion rules of let in statement, propose a condition extraction rules and action extraction rules of let in statement, and propose truth table generation rules of let in statement.

We confirmed that the improved VDTTable is applied to the triangle judgment VDM ++ specification including a let in statement to correctly generate the decision table. From the above, it can be said that the application range of the VDTTable improved in this paper has expanded, and its practicality has improved.

Future issues are as follows.

- Extension of scope
- Dealing with compound conditional expressions
- Improvement of readability
- Correspondence to functions including multiple syntax

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number 24220001

References


© The 2017 International Conference on Artificial Life and Robotics (ICAROB 2017), Jan. 19-22, Seagia Convention Center, Miyazaki, Japan