Classifications and Feature Analysis Method for Sort Algorithm Programs based on Program Structure Formalized Method

KISHIMOTO YORINORI
Department of Information Systems
Tokyo University of Information Sciences
4-1 Onari-dai, Wakaba-ku, Chiba
JAPAN
kisimoto@rsch.tuis.ac.jp

Abstract: - A learning of an algorithm is difficult for a programming beginner. When a programming beginner checks the program of the same algorithm from books, a website, etc. as a reference, which it is a source code from which they differ will confuse them. If the feature analysis in not an algorithm concept but an example program can be performed, it will become an aid of a beginner's algorithm teaching. Then, we focus on the feature analysis method of the algorithm by the program structure formalized method. In this paper, we propose the classifications method focused on the correspondence between a program structure and its specifications. In addition, we also propose the feature analysis method of a program mechanism. As a result of applying these two analysis methods to a practical example, it was shown that a bubble sort program and an insertion sort program have high specifications traceability. In addition, it was shown that a bubble sort program and a selecting sort program have a simple program mechanism.

Key-Words: - Software Engineering, Formal Approach, Static Analysis, Algorithms, Program Structure, Algorithms Education

1 Introduction
A learning of an algorithm is difficult for a programming beginner. Even if it is the program made from common specifications, the source codes may differ for some originators. Therefore, when a programming beginner checks programs in the same algorithm from books, a website, etc. as a reference, that it is a source code from which they differ will confuse them. We consider it one of the reasons that make the algorithm learning difficult.

In many cases, even if it is the program made from common specifications, they differ by each implementer's experience and specifications interpretation. Even if an input-output of a program is the same, these are implemented as a different program. For example, the difference in a variable name, and the difference of using a "break statement" at iterations, even if it is a simple function such as a swap of a value, a program changes with differences in an implementation, layout of process elements, etc. Such as these examples, the surface difference in a program makes unclear the difference of algorithm in a source code. Accordingly, an analysis of the source code which removed these differences can express the difference in an algorithm clearly.

Then, we focused on the program structure formalized method [1][2][3]. In this idea, a program is divided into a program structure and a program mechanism, and it analyzes focusing on the program structure. A program structure means maximum framework of the program. And program mechanisms are gives deterministic to a program structure. The program structure extracted from the program can be shown as a regular expression. By this abstraction, an analysis of the program which disregarded the surface difference such as "for statement" and "while statement", the difference in a variable name, etc. In addition, the correspondence analysis with a program and its specifications can be performed using this formula. Furthermore, it can analyze also about the similarity and the difference in a program which were made from common specifications using this formula.

Based on this idea, the surface difference in a program can be disregarded, and a program can be compared, and it can discuss also about a structural difference. Furthermore, at a beginners' class teaching, an algorithm sample easy to understand can be selected by the correspondence analysis with specifications.
In this paper, we propose the method of classifying
the source code of the sort algorithms based on the
program structure formalized method. In addition, we
apply this method to the sort algorithm which
appeared in books, and discuss a classification of these
algorithms, and each feature analysis.

2. Basic idea

2.1 Program structure formalized method

The construct of a program can be divided into the
maximum framework of a program (program
structure), and a control of the flow that gives a
deterministic (program mechanism). Program
mechanism is process elements which define a
program flow such as the proposition of selections
and/or the conditions of iterations.

A program structure is the maximum framework
of a non-deterministic process extracted from a
program by neglecting the program mechanism.
Three kinds of operators in regular expression can be
used to represent the program construct sequence; •
(AND) as concatenation, + (OR) as selection, and
Kleene Closures (()*,()+) as iterations.

A program structure cannot be discussed about the
stringency of a program. However, it can be discussed
about a program structure to discussion on the
framework of a program, and the correspondence
analysis between a program and specifications using
formula transformation.

2.2 Classification Idea

Based on the program structure formalized method, a
program consists of a program structure which is the
maximum framework, and program mechanisms give
deterministic to a program structure. Thus, the
differences in the program derived from common
specifications are divided into the difference in a
program structure, and the difference in program
mechanisms.

Because a program structure is a framework of a
program, the difference in a program appears in a
program structure first. And it implements as a
different program by giving a different program
mechanism to a common program structure.

In addition, the program structure can be analyzing
correspondence between a program structure and its
specifications. The program derived from common
specifications has a corresponding process element to
commom specifications. Thus, program elements
corresponding to its specifications that show the same
alphabet on a regular expression can be analyzed
similarity of a program structure.

It analyzes about a program structure first. Similarity of a framework design can be analyzed by
the grouping of the same program structure. The
difference in an algorithm can be checked by
analyzing the program mechanism given to these
program structures.

By the program structure formalized method, even
if the program structure elements unconnected to
discussion are ignored, it can analyze about the feature
of a program structure. The feature of the program
derived from common specifications can be classified
from the feature of each program mechanisms based
on same program structure.

3. Analysis method

This method consists of the following steps.

1. Reshaping of a program
At the conditions of selections and/or iterations, a
count and a process such as a call of a subprogram may
be included. In this case, a program is corrected so that
the conditions of selections and/or iterations may
become only a logical operation.

2. Separation of a program structure element
A program structure element is extracted from a
source code. The conditions of iterations in a program
and the conditions of selections are ignored.

3. Program structure formalized
Variables corresponding with the regular expression
are assigned to each program elements. A regular
expression shows a program structure based on these
variables and three kinds of operators in regular
expression can be used to represent the program
construct sequence; • (AND) as concatenation, +
(OR) as selection, and Kleene Closures (()*,()+) as
iterations.

4. Specifications element extract
The functional portion in a program is extracted from
specifications. Variables are given to the function of
specifications which corresponding to a regular
expression at a traceability analysis.

5. Traceability analysis
Variable names of a program structure that
corresponding to its specifications is replaced by the
variable name that extracted by the specification elements extraction.

(6) Classification of program structure
At a program structure, the regular expression variable which does not correspond to specifications is replaced by "ε", and these are classified.

(7) Classification of a program mechanism
Based on a program structure classification result, each formula is analyzed on the basis of program mechanisms.

4. Application Example
It shows application example of this method.

(1) Modification of a program
At the conditions of selections and/or iterations, a count and a process such as a call of a subprogram may be included. In this case, a program is corrected so that the conditions of selections and/or iterations may become only a logical operation. This example is shown in Fig. 1[4]. This example has the increment is included in the control of the "for statement". So, this is rewritten in a "while statement."

(2) Separation of a program structure element
A program structure element is extracted from a source code. The conditions of iterations in a program and the conditions of selections are ignored. An example of an extraction of a program structure is shown in Fig. 2. In Fig. 2, the part of the program mechanism is removed by making a proposition blank at iterations and a selection.

(3) Program structure formalized
Variables corresponding with the regular expression are assigned to each program elements. It shows in Fig. 2. And a regular expression derived from Fig.2 as follows;

\[
\text{a}\_b\_c\_1(c\_1(c\_2(c\_3+\varepsilon)c\_d)c\_b)^*}
\]

(4) Specifications element extract
The functional portion in a program is extracted from specifications. Variables are given to the function corresponding to a regular expression at a traceability analysis. A function of "exchange of values" derived from specifications because these examples are sort algorithm programs. So, variable name C was given to this function.

(5) Traceability analysis
The function of specifications and the variable name of a corresponding program structure are replaced by the variable obtained by the specifications element extraction. In this example, exchange of values C in a program is implemented by three elements (the interim storage of a value (temp=a[j]), the copy of a value (a[j]=a[j-1]), and the copy of a keeping value (a[j-1]=temp)). Then, these processes are set to \(C_1, C_2\), and \(C_3\) as follows;

\[
a\_b\_a\_C_1(C_1C_2C_3+\varepsilon)c\_d)c\_b)^*}
\]
Recent Researches in Software Engineering, Parallel and Distributed Systems

Table 1. Program Structure of Sort Programs appeared in books

<table>
<thead>
<tr>
<th>Algorithm Type</th>
<th>Program structure with mechanism elements</th>
<th>Variable does not correspond to specifications is replaced by $\varepsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bubble Sort[4]</td>
<td>$a_1 b_1 (c_1 (C_1 C_2 + \varepsilon) C_3 b_1)^*$</td>
<td>$((C_1 C_2 C_3 + \varepsilon)^*)$</td>
</tr>
<tr>
<td>Bubble Sort[5]</td>
<td>$a_1 b_1 (c_1 (C_1 C_2 + \varepsilon) a_1)^*$</td>
<td>$((C_1 C_2 C_3 + \varepsilon)^*)$</td>
</tr>
<tr>
<td>Bubble Sort[6]</td>
<td>$a_1 b_1 (c_1 (C_1 C_2 + \varepsilon) a_1)^*$</td>
<td>$((C_1 C_2 C_3 + \varepsilon)^*)$</td>
</tr>
<tr>
<td>Bubble Sort[7]</td>
<td>$a_1 b_1 (c_1 (C_1 C_2 + \varepsilon) a_1)^*$</td>
<td>$((C_1 C_2 C_3 + \varepsilon)^*)$</td>
</tr>
<tr>
<td>Bubble Sort[8]</td>
<td>$a_1 b_1 (c_1 (C_1 C_2 + \varepsilon) a_1)^*$</td>
<td>$((C_1 C_2 C_3 + \varepsilon)^*)$</td>
</tr>
<tr>
<td>Selecting Sort[4]</td>
<td>$a_1 b_1 (c_1 d_1 e_1 ((c_2 d_2 + \varepsilon) C_3 b_1)^*$</td>
<td>$((\varepsilon + \varepsilon)^* C_1 C_2 C_3)^*$</td>
</tr>
<tr>
<td>Selecting Sort[5]</td>
<td>$a_1 (C_1 b_1 C_2 + \varepsilon) C_3 a_1)^*$</td>
<td>$((\varepsilon + \varepsilon)^* C_1 C_2 C_3)^*$</td>
</tr>
<tr>
<td>Selecting Sort[6]</td>
<td>$a_1 (C_1 b_1 C_2 + \varepsilon) C_3 a_1)^*$</td>
<td>$((\varepsilon + \varepsilon)^* C_1 C_2 C_3)^*$</td>
</tr>
<tr>
<td>Selecting Sort[7]</td>
<td>$a_1 (C_1 b_1 (C_1 + j) a_1)^*$</td>
<td>$((\varepsilon + \varepsilon)^* C_1 C_2 C_3)^*$</td>
</tr>
<tr>
<td>Insertion Sort[4]</td>
<td>$a_1 b_1 (c_1 (C_1 C_2 + \varepsilon) a_1)^*$</td>
<td>$((C_1 C_2 C_3 + \varepsilon)^*)$</td>
</tr>
<tr>
<td>Insertion Sort[5]</td>
<td>$a_1 (C_1 b_1 C_2 + \varepsilon) C_3 a_1)^*$</td>
<td>$((C_1 C_2 C_3 + \varepsilon)^*)$</td>
</tr>
<tr>
<td>Insertion Sort[6]</td>
<td>$a_1 (C_1 b_1 C_2 + \varepsilon) C_3 a_1)^*$</td>
<td>$((C_1 C_2 C_3 + \varepsilon)^*)$</td>
</tr>
<tr>
<td>Insertion Sort[7]</td>
<td>$a_1 (C_1 b_1 (C_1 + j) a_1)^*$</td>
<td>$((C_1 C_2 C_3 + \varepsilon)^*)$</td>
</tr>
<tr>
<td>Insertion Sort[8]</td>
<td>$a_1 (C_1 b_1 C_2 + \varepsilon) C_3 a_1)^*$</td>
<td>$((C_1 C_2 C_3 + \varepsilon)^*)$</td>
</tr>
<tr>
<td>Shell Sort[4]</td>
<td>$a_1 b_1 (b_2) (c_1 d_1 (C_1 C_2 C_3 d_2) C_3 b_1)^*$</td>
<td>$((\varepsilon)^* (((C_1 C_2 C_3 + \varepsilon)^<em>)^</em>)$</td>
</tr>
<tr>
<td>Shell Sort[5]</td>
<td>$a_1 (a_2) (b_2) (c_1 d_1 (C_1 C_2 C_3 d_2) C_3 b_1)^*$</td>
<td>$((\varepsilon)^* (((C_1 C_2 C_3 + \varepsilon)^<em>)^</em>)$</td>
</tr>
<tr>
<td>Shell Sort[6]</td>
<td>$a_1 (a_2) (b_2) (c_1 d_1 (C_1 C_2 C_3 d_2) C_3 b_1)^*$</td>
<td>$((\varepsilon)^* (((C_1 C_2 C_3 + \varepsilon)^<em>)^</em>)$</td>
</tr>
<tr>
<td>Shell Sort[7]</td>
<td>$a_1 (a_2) (b_2) (c_1 d_1 (C_1 C_2 C_3 d_2) C_3 b_1)^*$</td>
<td>$((\varepsilon)^* (((C_1 C_2 C_3 + \varepsilon)^<em>)^</em>)$</td>
</tr>
<tr>
<td>Shell Sort[8]</td>
<td>$a_1 (a_2) (b_2) (c_1 d_1 (C_1 C_2 C_3 d_2) C_3 b_1)^*$</td>
<td>$((\varepsilon)^* (((C_1 C_2 C_3 + \varepsilon)^<em>)^</em>)$</td>
</tr>
</tbody>
</table>

(6) A classification of a program structure
At a program structure, the regular expression variable which does not correspond to specifications is replaced by $\varepsilon$, and these are classified. The application result of this method to the practical examples in books are shown in Table 1 [4][5][6][7][8].

(7) Classification of a program mechanism
Based on the program structure classification result, each formula is analyzed on the basis of program mechanisms. Regular expressions that include program mechanisms of this example are shown in Table 1. In this example, in order to discuss simply, the subscript of the following was given to the variable of the regular expression.

Increment is "i"
Decrement is "d"
Set of a value is "s"

In addition, the same statement has given the same variable.

5. Consideration
5.1 Classification by program structure
From Table 1, it focuses on value exchange ($C_1$, $C_2$, $C_3$) as a feature of a program structure at sort programs. In this example, a process of $C_1 C_2 C_3$ can be classified, in cases where it implements to a concatenate and in cases where it implements individually. Bubble sort programs and selecting sort programs tend to be implemented by making these concatenated. Insertion sort programs and shell sort programs tend to be implemented by making these separated.

In addition, the basic program structure was derived from each sort program. It shows in Table 2.

Table 2. Basic program structure of Sort program

<table>
<thead>
<tr>
<th>Algorithm Type</th>
<th>Basic program structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bubble Sort</td>
<td>$((C_1 C_2 C_3 + \varepsilon)^*)$</td>
</tr>
<tr>
<td>Selecting Sort</td>
<td>$((\varepsilon + \varepsilon)^* C_1 C_2 C_3)^*$</td>
</tr>
<tr>
<td>Insertion Sort</td>
<td>$((C_1 C_2 C_3 + \varepsilon)^*)$</td>
</tr>
<tr>
<td>Shell Sort</td>
<td>$((\varepsilon)^* (((C_1 C_2 C_3 + \varepsilon)^<em>)^</em>)$</td>
</tr>
</tbody>
</table>

The feature of bubble sort program is $C_1 C_2 C_3$ into a double nesting iteration. The feature of selecting sort program is skip process before $C_1 C_2 C_3$ into the
iteration of a double nesting. The feature of insertion sort program is iterating only $C_2$. The feature of shell sort program is skip process of iteration before a ternary nesting iteration. In this example, the bubble sort program and the selecting sort program were high specifications traceability because $C_1C_2C_3$ implemented by concatenation.

5.2 Feature of program mechanism
We focused on the program mechanisms. At bubble sort programs and insertion sort programs, it shown that increments and decrements are main program mechanisms. It was shown that selecting sort programs has assignment of a variable in part of program mechanisms. It was shown that a shell sort program has much assignment of a variable, and this program mechanism process is complicated. In this example, the bubble sort program and the insertion sort program were simple program mechanisms.

5.3 Application in teaching
The feature of the sort program of this example is shown in Table 3.

Table 3. Feature of sort algorithm program

<table>
<thead>
<tr>
<th>Algorithm Type</th>
<th>Specification</th>
<th>Program Mechanism</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bubble Sort</td>
<td>High</td>
<td>Simple</td>
</tr>
<tr>
<td>Selecting Sort</td>
<td>High</td>
<td>Difficult</td>
</tr>
<tr>
<td>Insertion Sort</td>
<td>Low</td>
<td>Simple</td>
</tr>
<tr>
<td>Shell Sort</td>
<td>Low</td>
<td>Difficult</td>
</tr>
</tbody>
</table>

At these examples, it is considered that the bubble sort program is better sort algorithm for beginner education, because those have the high specifications traceability and the simple program mechanism.

In addition, at the teaching of a selecting sort program, we consider that instruction can be made smooth by showing the difference of a program mechanism between a selecting sort program and a bubble sort program.

Furthermore, at the teaching of an insertion sort program, we consider that instruction can be made smooth by showing the difference of a program structure between an insertion sort program and a bubble sort program.

6. Conclusion
In this paper, we proposed the application of the program structure formalized method for the feature analysis of sort programs.

As a result of applying this method to a practical example, it was shown that a bubble sort program and an insertion sort program have high specifications traceability. In addition, it was shown that a bubble sort program and a selecting sort program have a simple program mechanism.

References: