A Programming Education Support Tool pgtracer utilizing Fill-in-the-Blank Questions

Teacher Functions

Tetsuro Kakeshita
Ryo Yanagita
Kosuke Ohta
Mika Ohtsuki
Graduate School of Information Science
Saga University, Saga 840-8502, Japan

***Abstract*— We are developing a programming education support tool pgtracer utilizing fill-in-the-blank questions. Pgtracer operates on Moodle, and provides questions to the student that some parts are masked in a pair of program and trace table. A question consists of four XML files representing a program, a trace table and two types of masks for program and trace table. We propose teacher's function of pgtracer in this paper: question editing and data analysis functions. The question editing function includes automatic generation function of XML files representing program and trace table from a given C++ source program and input data. Pgtracer collects student's log and provides data analysis function from the various viewpoints of student and question.**

Keywords-computer programming; programming education; fill-in-the-blank question; collection and analysis of student log; online education; education support tool

I. INTRODUCTION

Programming education is important at university and high school particularly majored in science and engineering. However there are obstacles in programming education since average student's ability is declining due to the increase of the number of students at higher education. The lack of support staff for programming education is also observed frequently.

We are developing a programming education support tool named pgtracer utilizing fill-in-the-blank questions [1]. Pgtracer operates on a widely accepted lecture management system Moodle [2] as a plug-in module. Thus a student can use pgtracer 24 hours 365 days even from outside the school. The instructor can distribute various educational contents from a Moodle course page.

Pgtracer provides a fill-in-the-blank question to the students (Figure 1). A question is composed of a pair of a program and a trace table. The trace table represents execution order of steps and the values of the variables and output at each step of the program. An instructor can define a blank at an arbitrary place of the question such as single token, expression, compound statement and entire routine of a program as well as variable value, variable name and step number of a trace table. Thus an instructor can control the difficulty level of a question by utilizing various types of blanks as explained above. Consequently pgtracer can be adapted to a wide range of students with various achievement levels.

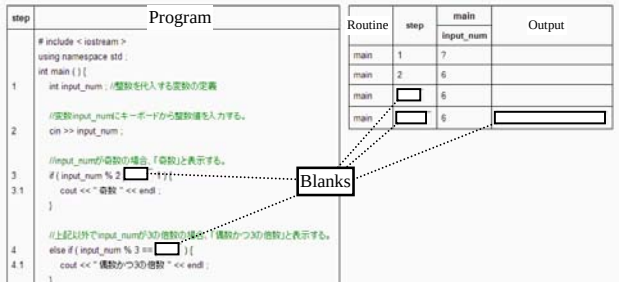


FIGURE I. FILL-IN-THE-BLANK QUESTION OF PGTRACER

We propose the teacher function of pgtracer in this paper. The function is categorized into question editing function and data analysis function. The question editing function is utilized to generate four types of XML files representing a question. Pgtracer collects student's log when a student fills a single blank. The data analysis function is utilized to understand the achievement level of each student as well as the achievement level of the entire class for each question. An instructor can also review the answering process of a student to improve the educational contents and the instruction to the student.

II. XML DESIGN FOR FILL-IN-THE-BLANK QUESTIONS

Pgtracer represents a question by four types of XML files. This is because a single program can have multiple trace tables depending on the input values. An instructor may define multiple masks on a single program or trace table in order to control difficulty level of the question.

A. Program Description

XML file representing a source program contains structure to describe program, comment, class, routine, definition, compound statement, simple statement, token as XML tags as well as correspondence between step number and statements. The XML file can describe programs written in C++ and Java. The "definition" tag is used to represent definition of a variable or a data type. The "compound statement" tag is composed of more than one "simple statement". A step corresponds to either a simple statement or a compound statement. In either case, the statement is specified using an XPath expression.

B. *Trace Table*

A trace table is a table whose row corresponds to a step and whose column corresponds to a variable defined in the program. Thus we designed the XML definition similar to the table description in html.

C. *Program Mask*

Pgtracer allows defining multiple masks for a program. Figure II represents the DTD for program mask. Each mask is defined by a "question" element and position of the mask within the corresponding program. The position of a mask is described using an XPath expression. Thus an arbitrary sequence of tokens within the program can be described.

```
<!DOCTYPE mask-for-program [
 <!ELEMENT mask-for-program (
   hidden*, question*)>
  <!ATTLIST mask-for-program
    id CDATA #REQUIRED
    target-program CDATA #REQUIRED>
 <!ELEMENT hidden EMPTY>
  <!ATTLIST hidden
    target-path CDATA #REQUIRED>
 <!ELEMENT question EMPTY>
  <!ATTLIST question
   target-path CDATA #REQUIRED
   weight CDATA "1">
]>
```

FIGURE II. DTD FOR PROGRAM MASK

The "hidden" element of the DTD is introduced not to show a specific portion of the program, such as comment, for the question. Difficulty level of a fill-in-the-blank question can also be controlled by utilizing the "hidden" element.

The "target-program" attribute is used to specify XML file names of the target program.

D. *Trace Table Mask*

The trace table mask represents the position of masks by using "question" element of the DTD illustrated in Figure III. As in the case of the DTD for program mask, the position of the mask in the corresponding trace table is defined using an XPath expression.

```
<!DOCTYPE mask-for-trace-table [
 <!ELEMENT mask-for-trace-table (
   schema, row*, question*)>
  <!ATTLIST mask-for-trace-table
    id CDATA #REQUIRED
    target-trace-table CDATA #REQUIRED>
 <!ELEMENT schema (
   step-number-header, variable-definition*)>
 <!ELEMENT step-number-header (#PCDATA)>
 <!ELEMENT variable-definition EMPTY>
  <!ATTLIST variable-definition
    target-path CDATA #REQUIRED>
 <!ELEMENT row EMPTY>
  <!ATTLIST row target-path CDATA #REQUIRED>
 <!ELEMENT question EMPTY>
  <!ATTLIST question
    target-path CDATA #REQUIRED
    weight CDATA "1">
]>
```

FIGURE III. DTD FOR TRACE TABLE MASK

The row element is used to specify a set of trace table rows visible to the students. The row element can be used conveniently when a trace table contains a loop and thus becomes long and complicated.

The "variable definition" element can be used to change the position of a variable within the trace table. It can also be used to hide a variable. Such definition becomes possible by describing the element within the "schema" element. This can also control the difficulty level of a question.

III. PGTRACER FUNCTIONS FOR TEACHERS

Teacher's function of pgtracer is categorized in two types. The question editing function corresponds to four types of the XML files. Once the four types of XML files are prepared, the instructor can define a question by specifying a valid combination of the files. The data analysis function is composed of the analysis of question, student, student's answering process and each blank of a question.

A. *Automatic Generation of XML Files for Programs*

An instructor can upload a C++ source program to pgtracer. Pgtracer checks the program through compilation and converts the source program to an XML file explained in Section II.A. The left and right sides of Figure IV represents the generated XML file and the original C++ program respectively.

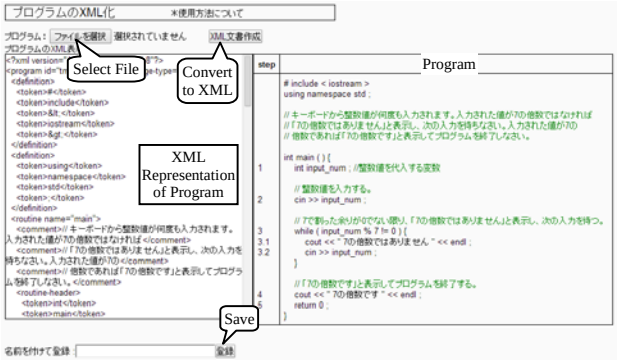


FIGURE IV. GENERATION OF XML FILE FOR PROGRAM

Current versition of pgtracer generates XML file containing the following C++ grammer and library functions.

- Global and local variables, constant
- Data types (int, long, double, float, char, char*, char[], String, bool, pointer)
- Data structure (array, record)
- Expression (arithmetic and logical expressions, comparison, assignment)
- String manipulation (strcat, strcmp, strlen, strcpy)
- I/O (cin, cout, printf, scanf, ofstream, ifstream)
- Control statement (if, else, switch, while, for, do-while)
- Function definition, parameter, function call, return

B. *Automatic Generation of XML Files for Trace Table*

Pgtracer automatically generates XML file for trace table from the XML file describing program and a text file supplying input data for the program. When an instructor selects an XML file for a program, pgtracer shows the corresponding C++ program as illustrated in Figure V. Pgtracer supports standard input and

file input for the program. The instructor next specifies a file name describing the input data for the both cases. Pgtracer then execute the program using the input data and shows the execution result. When the instructor confirms the execution result, pgtracer automatically generates XML file representing the trace table corresponding to the execution result.

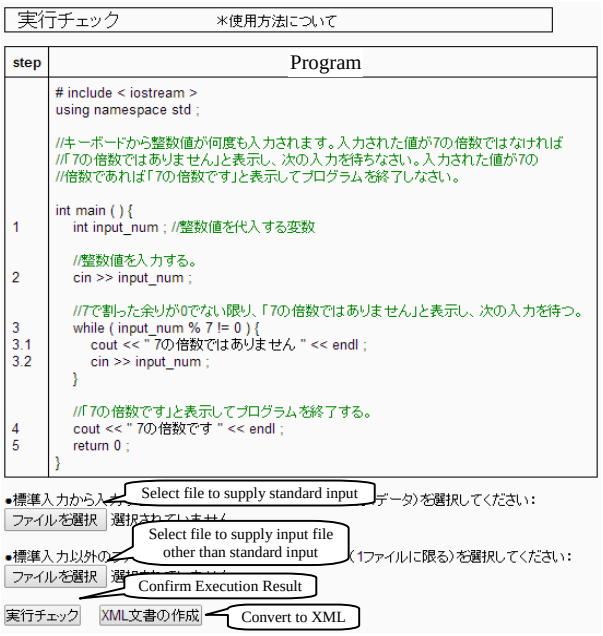


FIGURE V. GENERATION OF XML FILE FOR TRACE TABLE

Pgtracer provides automatic generation function of XML files representing program and trace table so that an instructor can easily generate XML file even if he does not have knowledge and experience on XML language.

C. *Creation and Editing of XML Files for Program Mask*

In order to create and edit XML files for program mask and trace table mask, pgtracer provides editing function so that an instructor can use without XML knowledge and experience.

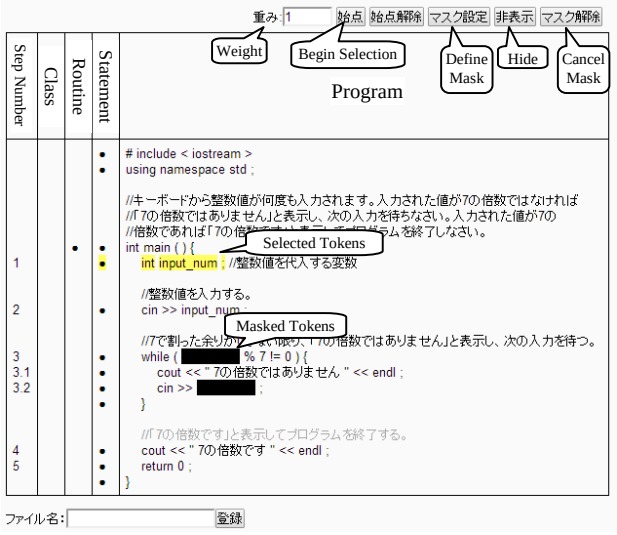


FIGURE VI. EDITING OF XML FILE FOR PROGRAM MASK

Figure VI illustrates the user interface to edit program mask. When an instructor selects an XML file representing a program, pgtracer shows the corresponding program. Then the instructor can define masks, hidden tokens and weight of each mask on this user interface. The instructor can specify an arbitrary sequence of tokens by using “Begin Selection” and “Define Mask” buttons. He can also specify hidden tokens in a similar manner. Pgtracer also provides the following functions for convenient editing of the program mask.

- Defining weight of a mask
- Cancellation of defined mask and hidden tokens
- Selection of entire statement, single routine and a class in order to mask or hide them
- Editing of a stored XML file representing a program mask

D. *Creation and Editing of XML Files for Trace Table Mask*

Figure VII illustrates the user interface to edit trace table mask. When an instructor selects an XML file representing a trace table, pgtracer shows the corresponding trace table. Then the instructor can define masks on this user interface. The editing function also provides the following functions for the instructor.

- Defining weight of a mask
- Defining hidden row
- Cancellation of defined mask and hidden row
- Selection of multiple fields contained in a specified rectangle within the trace table
- Exchange of the columns representing variables
- Editing of a stored XML file representing a trace table mask

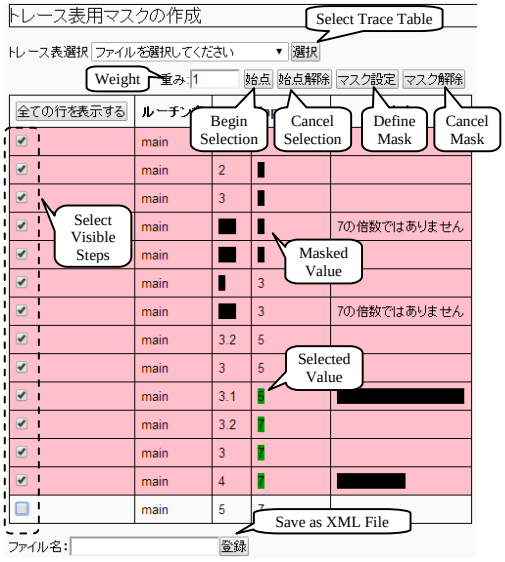


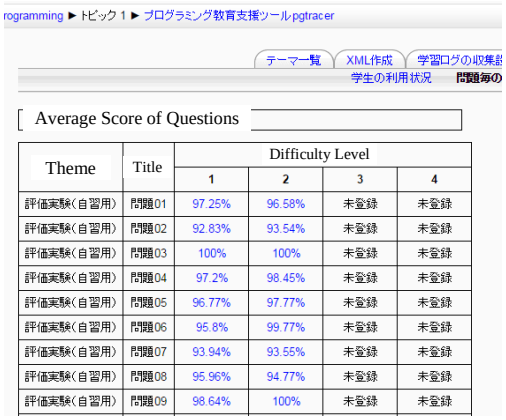
FIGURE VII. EDITING OF XML FILE FOR TRACE TABLE MASK

E. *Analysis of Questions*

In addition to the editing function of various XML files, pgtracer provides data analysis function. Pgtracer can collect study log and answer log while students solve fill-in-the-blank questions. The study log accumulates logs containing overall score of the answers and is collected when a student answers a question. The answer log contains individual score and answer of each blank and is collected when a student fills a blank. The

data analysis function provides the following result by analyzing the logs.

- Overview analysis and learning history of student
- Overviewl analysis and learning history of question
- Analysis of answering process of a student
- Detailed analysis of each blank of a question



Theme	Title	Difficulty Level			
		1	2	3	4
評価実験(自習用)	問題01	97.25%	96.58%	未登録	未登録
評価実験(自習用)	問題02	92.83%	93.54%	未登録	未登録
評価実験(自習用)	問題03	100%	100%	未登録	未登録
評価実験(自習用)	問題04	97.2%	98.45%	未登録	未登録
評価実験(自習用)	問題05	96.77%	97.77%	未登録	未登録
評価実験(自習用)	問題06	95.8%	99.77%	未登録	未登録
評価実験(自習用)	問題07	93.94%	93.55%	未登録	未登録
評価実験(自習用)	問題08	95.96%	94.77%	未登録	未登録
評価実験(自習用)	問題09	98.64%	100%	未登録	未登録

FIGURE VIII. OVERVIEW ANALYSIS OF QUESTIONS

Figure VIII illustrates the overview analysis function of questions. It provides average score of each question. There is a case that a student solves a question more than once. In such a case, the maximum score is used to calculate the average score. The questions are classified using theme, title and difficulty level. The average score is also used as a link to the learning history analysis of the corresponding question.

Pgtracer also provides similar overview analysis function of the students.

F. *Analysis of Each Student*

Figure IX illustrates the learning history of a student. The learning activity of each student can be observed using the function. Pgtracer also provides similar analysis function of each question.

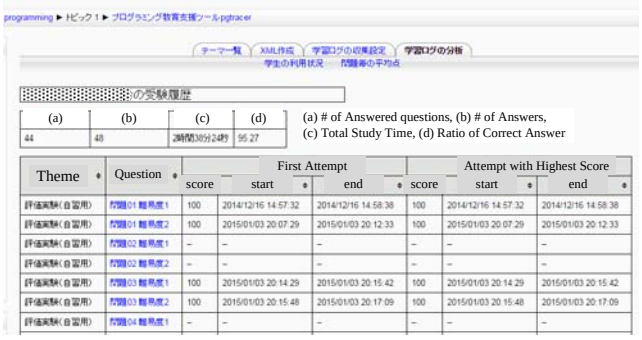


FIGURE IX. LEARNING HISTORY ANALYSIS OF A STUDENT

G. *Analysis of Answering Process of a Student*

Figure X illustrates the answering process of a student for a question. An instructor can view intermediate states of the answer by selecting a radio button corresponding to each log record of the answer log.

FIGURE X. ANSWERING PROCESS OF A STUDENT

H. *Detailed Analysis of Each Blank of a Question*

Figure XI illustrates the analysis of each blank of a question. An instructor can select an arbitrary blank of a question to view all the answers of the students to the selected blank.

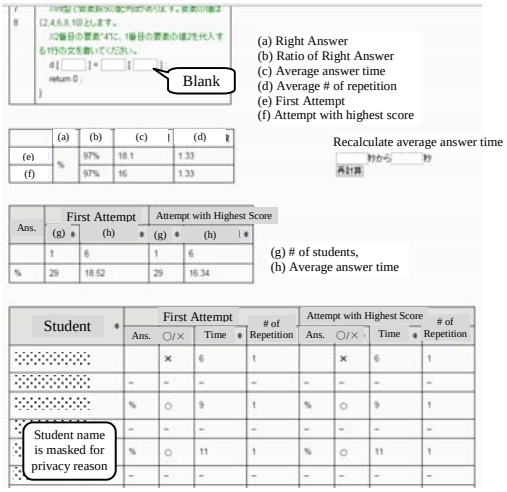


FIGURE XI. ANALYSIS OF EACH BLANK OF A QUESTION

IV. CONCLUSION

We have developed teacher's function of pgtracer in this paper. Although we omit the detailed explanation due to space limitation, we have asked for some professors in charge of programming courses at Saga University and Kumamoto National College of Technology to use these functions. We have received positive comments concerning the usability of the function.

We are planning to perform a more complete evaluation of the teacher's function of pgtracer. We are also working to integrate graphs to the data analysis function in order to show the analysis result more effectively. Then an instructor quickly understands the achievement level of each student and the entire class and can improve the programming education.

REFERENCES

[1] T. Kakeshita, et al., "A programming education support tool pgtracer utilizing fill-in-the-blank uestions: overview and student functions", Proc. 2nd International Conf. on Education Reform and Modern Management (ERMM 2015), Atlantis Press, April 2015, 4 pages. (in Press)