Automatic Assessment of Programming assignment

Surendra Gupta[1] and Shiv Kumar Dubey[2]

Department of Computer Engineering
Shri G. S. Institute of Technology & Science 23, Park Road Indore 452003 (MP) India

[1]sgupta@sgsits.ac.in, [2]shivmgcgv@gmail.com

ABSTRACT

In today's world study of computer's language is more important. Effective and good programming skills are need full all computer science students. They can be master in programming, only through intensive exercise practices. Due to day by day increasing number of students in the class, the assessment of programming exercises leads to extensive workload for teacher/instructor, particularly if it has to be carried out manually.

In this paper, we propose an automatic assessment system for programming assignments, using verification program with random inputs. One of the most important properties of a program is that, it carries out its intended function. The intended function of a program or part of a program can be verified by using inverse function's verification program. For checking intended functionality and evaluation of a program, we have used verification program. This assessment system has been tested on basic C programming courses, and results shows that it can work well in basic programming exercises, with some initial promising results.

KEYWORDS

Programming assignment, automatic assessment, verification program, input/output file

1. INTRODUCTION

As stated in ACM Computing Curricula 2001 [1], programming-involved courses are regarded as the basis of most of the computer science studies. In the other words, possession of good programming skill is necessary to secure the learning outcomes in this field. In the past and in the present time also, at many places programming is mostly taught in the traditional face-to-face basis. When faced with grading hundreds of program manually only the help of teaching assistants is not so easy. In large class error detection, manual evaluation for every program is difficult and time consuming. With the recent advancement of the internet technologies and advanced program analysis techniques, web-based tutoring systems that can play the role of teacher for teaching and training programming, partially or completely, are increasingly considered [2]. Automatic assessment of programming assignment is one major task in programming classes to evaluate and marking students' programming exercises aided by the computer. For the mastery of programming skill, solid background in theory and profound experience in practical exercises are both required [3]. To render learners with a convenient environment to observe theoretical issues in programming, educational tool like Alice [4] has tried to employ visual effects and animation to illustrate the programming concepts. In traditional education, this problem is countered by manual review and inspection offered by class tutors [5].

DOI : 10.5121/cseij.2012.2107

However, it is not efficiently practical when dealing with large numbers of students and practice exercises. To automate the inspection process, one popular method suggested is generating a suit of test cases based on the coverage analysis [6] and executing the programs with all of the test cases. Even though applied widely in industry, this method is not theoretically complete since test cases are impossible to secure the absence of possible flaws in programs. In addition, it is not always encouraged to run a piece of potentially erroneous code due to the system safety. With the recent advancement of the internet technologies and advanced program analysis techniques, web-based tutoring systems that can play the role of teacher for training programming, partially or completely, are increasingly considered. In this paper a system is proposed, which can help and intended to improve education performance in the course of programming methodologies.

The rest of the paper is organized as follows. Section II, presents some related works in this field. Section III, presents assignment assessment processing, in which we discuss about verification program and assessment approach concepts. In next section V, presents our proposed system. The second last section VI, presents experimental results by the system. Finally, section VII, concludes the paper with some future directions pointed out.

2. RELATED WORKS

Research in the context of automatic programming assessment has a long history. It has been of interest to computer science educators started from 1960s and has continued to gain vast attention till present. Its core aims are mainly to promote an automated tool to reduce the workload of human teachers, to improve consistency of marking assessment items and to include thorough testing of students' programming exercises. The basic requirement for automated assessment of programming exercises are the measurement values that can be extracted from the program and the values can be compared to the given requirements or to a model solution. For an educational purpose, the measurement values typically need to be justified by the teaching goals of the course or specifically by objectives to be achieved for each topic of the course syllabus. Several approaches to automatic programming assessment can be found from various resources such as journals, conference articles and online resources. The approaches typically based on either static analysis or dynamic testing. This refers to whether a program needs to be executed while it is being assessed. On top of that, the assessment process can be done by looking into a code structure (white-box) or simply based on a functional behavior of a program (black-box) [7].

A. Static program analysis: Static program analysis has recently emerged as an efficient means for program verification. There are two popular approaches arisen in this direction: theorem proving and model checking. The theorem proving approach, supported by the logic of axiomatic semantic theory, is sound and complete for correctness verification. However, due to the computational complexity, this approach fails to produce counter-examples of the errors encountered, thus limited in terms of education sake. Model checking approach can tackle this problem, but this method cannot work on a virtually infinite domain hence probably suffering infinitive execution when processing flaw-free programs [8].

Static analysis doesn't execute student programs, so it is feasible for programs that cannot be compiled successfully. Two steps are needed: transformation of student program and model program into an intermediate representation and analysis after transformation.

Yingli Liang et al [9] describe the transformation of student program into intermediate representation and analysis after transformation and how the static analysis can help in program verification. The transformation needs to be done in the following steps: First, an intermediate representation form should be selected and generated from the source code. A category of general intermediate representation includes: textual form such as characters and strings, Abstract Syntax

Tree (AST) and graphs such as Control Flow Graph (CFG), Program Dependence Graph (PDG) and System Dependence Graph (SDG). Second, standardization of intermediate representation should be done to reduce code diversity. Basic standardization includes temporary declaration standardization, expression standardization, control structure standardization and so on. The transformation needs to be done in the following steps: First, an intermediate representation form should be selected and generated from the source code. A category of general intermediate representation includes: textual form such as characters and strings, Abstract Syntax Tree (AST) and graphs such as Control Flow Graph (CFG), Program Dependence Graph (PDG) and System Dependence Graph (SDG). Second, standardization of intermediate representation should be done to reduce code diversity. Basic standardization includes temporary declaration standardization, expression standardization, control structure standardization and so on.

The automated programming assessment using the pseudo-code comparison technique is one of the techniques suggested in the static analysis approach; which is the non-structural similarity analysis. This technique suggests that both the students' source code and the instructors' source-code are translated into their respective pseudo-code, before they are compared. The instructors' source-code will act as the answer scheme. Khirulnizam Abd Rahman at al [10] described an application which is developed to assess student C programming exercises based on the pseudocodes. According to author the purpose of developing this application is to find the percentage of the pseudocode similarity between student's answer and the instructor's scheme. The method used in their software is by translating the students' programming answer and all the instructors' answer schemes into pseudocode. The software will compare the students' pseudocode with all the pseudocode from the instructors' answer schemes. The highest percentage of similarities will be chosen for the mark.

C. A. R. Hoare [11] introduced Hoare logic in his paper. The central feature of Hoare logic is the Hoare triple. A triple describes how the execution of a piece of code changes the state of the computation. A Hoare triple is of the form {P} C {Q} where P and Q are assertions and C is a command. P is named the precondition and Q the postcondition, when the precondition is met, the command establishes the postcondition. Hoare logic provides axioms and inference rules for all the constructs of a simple imperative programming language.

B. Dynamic Program Analysis: For manual assessment, it is difficult to figure out the results of a program because the program may has many different executive paths and output results even it is simple. Dynamic analysis assesses a program by executing it based on the test data which are automatically generated or manually provided. In other word dynamic program analysis is the analysis of computer software that is performed by executing programs built from that software system on a real or virtual processor. For dynamic program analysis to be effective, the target program must be executed with sufficient test inputs to produce interesting behavior. Use of software testing techniques such as code coverage helps ensure that an adequate slice of the program's set of possible behaviors has been observed. Also, care must be taken to minimize the effect that instrumentation has on the execution (including temporal properties) of the target program.

The general dynamic analysis proceeds as followings: first, it is a must that the program be compiled successfully; second, it is necessary to take as many security threats as possible into consideration, such as infinite loops, fatal errors during execution as well as malicious codes. Existed methods to cope with these challenges include scanning the codes before executing the program, access permission or putting a limit on system resources and so on; third, the complete test data are required in execution. Test data could be obtained in two ways, manually and automatically generated by computer; finally the output of the program and information throughout the process should be collected and analyzed in an appropriate way [12].

3. MATHEMATICAL MODEL

All programs (let a program is P) are generated by the some algorithms (let the algorithms is A), and all algorithms are represented by function or the group of functions (let the function is F). And all functions will also have its inverse function (let inverse function is F^{-1}), now with this inverse function F^{-1}, we can write a program (let the program called P^{-1} which will have the reverse functioning of program P).

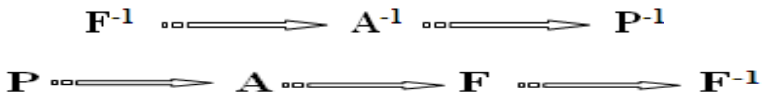

Figure 1. Steps of getting Program P^{-1} from Program P

Here in figure 1, lets P is a Program, A is an Algorithms, F is a function or group of functions for algorithm A and F^{-1} is an inverse function, with respect to function F. A^{-1} is an algorithm for function F^{-1} and P^{-1} is a program for algorithm A^{-1}. So in reverse manner we can write a program P^{-1}, this program can use to check the functioning of program P. Now if we can make a program P^{-1} from program P, which has the inverse functioning of program P, then we can assess and evaluate the program P.

If any programming problem which is given to student satisfies certain property, then with use of this we can make an inverse function for that problem. In other word if we can make a reverse program of a given programming problem from inverse function approach, then we can make a verification program for that particular problem and with this we can assess the assignment by this system.

Inverse function

If f maps X to Y, then f^{-1} maps Y back to X as shown in figure 2.

Figure 2. Inverse Function

Let f be a function whose domain is the set X, and whose range is the set Y. Then f is invertible if there exists a function g with domain Y and range X, with the property: f(x) =y if and only if g(y) =x. If f is invertible, the function g is unique; in other words, there can be at most one function g satisfying this property. That function g is then called the inverse of f, denoted by f^{-1}.

Example

A function is f and its inverse f^{-1}. Because f maps 'a' to 3, the inverse f^{-1} maps 3 back to 'a'.

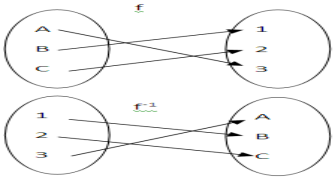

Figure 3. Example of Inverse Function

In mathematics, an inverse function is a function that in some sense undoes another function: If an input x into the function f produces an output y, then putting y into the inverse function g produces the output x, and vice versa. i.e., f(x) =y, and g(y) =x. More directly, g(f(x)) =x, meaning g(x) composed with f(x) leaves x unchanged. The inverse function approach is used to write a verification program for assess given assignment.

4. ASSIGNMENT ASSESSMENT PROCESSING

In our system, we employ the theory of inverse function property satisfy by solution of any problem, to handle the task of automatic program assessment. Assessment processing mainly done around following: problem description, student program, verification program.

Problem description

Teacher defines a problem as, (1) a natural description. (2) Verification program for that problem description. While the natural description is explicitly presented to the student, the verification program is hidden and only used by the system to assess the submissions, submitted by the student. For example:

A. Program Description

The problem of finding the quotient Q and remainder R obtain on dividing X by Y. All variables are assumed to range over a set of nonnegative integers. Lets the student submitted program's code part is:

```
R=X; Q=0;
do { R=R-Y; Q=Q+1; }
while (Y<=R)
```

Verification Program- An important property of this program is that when it terminates, we can recover the numerator X by adding to the remainder R the product of the divisor Y and the quotient Q (i.e. X=R+Y*Q). Furthermore, the remainder is less than the divisor. These properties may be expressed formally: Y<=R^X+Y*Q. Now teacher write a program which check whether or not condition is fulfil by program after execution, whether or not it carries out its intended function. Like if a condition X=R+Y*Q is satisfied after execution of student program then the program is correct otherwise incorrect.

B. Program Description

Program for find out the inverse of a matrix m

Verification Program

Let the inverse of matrix m given by student program is m^{-1}. Then the property $m * m^{-1} = [I]$ ([I] the identity matrix or unit matrix) should satisfy. For checking these above condition, we can write a verification program.

Now we have the: 1. Student program. 2. Verification program. 3. Input/output of student program (input for program random generated by system). 4. Now combination of input and output of student program will be the input for verification program as if figure 4.

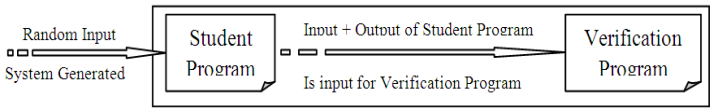

Figure 4. Input/output Approach

Consider if we have a specific condition for every specific problem, and if solution given by student is correct then it should be satisfy these condition.

That's why if we can write a verification program of any given problem with in inverse properties satisfied by given programming problem, then we can assess programming assignment through this approach. Here only need is to find out the specific inverse function of a given problem and write a verification program, through which we can assess bulk of programs.

5. PROPOSED SYSTEM

Here we are going to propose a system which provides the facilities for teacher to give the programming assignments to students. In which he can give the programming assignment description and its verification program and student will submit their assignments. System verifies the submitted programming exercises according to teacher instruction. This assessment process can help all computer study centre with respect to programming-involved courses like C programming language. It will help the teacher to promote the subject and increase the course performance and also increase the quality of understanding. By this system, students also increase the interest to study and understanding the concept of the programming subjects.

In this paper, we suggest an approach used verification program technique to automatic assessment programming assignments, without worry about the execution of program, input and getting output. Our system for automatic assessment of program exercises is given bellow in figure 5.

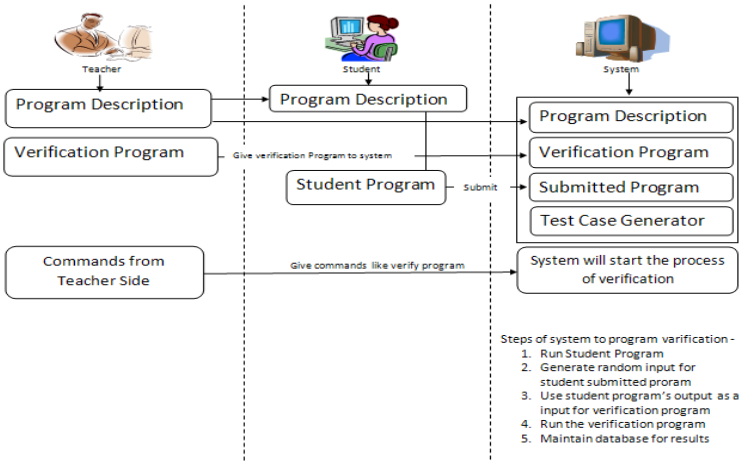

Figure 5. Proposed Assignment Assessment System

There are three actors working in this system: Teacher, Student and System. First, Teacher will provide programming problems (which are presented descriptive to student) and verification program. The programming problem will descriptive to student and verification program is hidden. Later when student visits the system, he can try to solve these problems. The works submitted by student is then assessed by System. The stochastic information of system, such as common errors or error of program, number of input, number of time run will store in system's database, which can helps teacher to evaluate the performance of the students and whole course.

5. EXPERIMENTAL RESULT

The system is implemented in course of basic programming concept. There were 90 students in class and each one have to submit assignment of three programming problems. Students assigned the roll number from 1 to 90 for differentiate among submitted assignments. Teacher gives the description of assignment and later, student will submit their assignment through mail.

Teacher gives mail to student, in which he gives detail description of assignment. The description like natural description of problem, due date of submission and some special instruction (like program name should in specific formant, program should without input scanf( ), getch( ) type input function and program output format etc) etc. Here is the result of assignment assessment through system in Table 1.

Table 1 System assessment summary

Number of program run	**Correct**	**Incorrect**	**Compile error**
81 programs run	55%	25%	20%

6. CONCLUSIONS AND FUTURE WORK

This "Automatic assessment system" automatically compiles and runs the students program and evaluate on the basis of verification program. The system proposed in this paper has been implemented in a Computer class based on basic programming concepts. There were 90 students in class and 81 students submitted their assignments. The assessment of the assignment is done by this system. In this system 81 students assignments evaluated, in which there was 20% compilation error, 25% incorrect programs and rest of 55% programs is resulting correct as per result of system. This system gives the result of assessment in less effort and provides all information regarding programming assessment.

The proposed system has some constraints, like student have to write a program which takes input only form command line argument and format of output should be in predefined format. In future these constraints can we remove to make system more user friendly. Till now this system can't we say as fully automated, in future by enhancing this system, we can make this, fully online and automatic which can be use from anywhere through internet.

REFERENCES

[1] The Joint Task Force: Computing Curricula 2001, Computer Science, ACM, 2001.
[2] Tho T. Quan, Phung H. Nguyen, Thang H. Bui, Linh V. Huynh and Anh T. A framework for automatic verification of programming exercises.

[3] H. Dobler, R. Ramler and K. Wolfmaier. A Study of Tool Support for the Evaluation of Programming Exercises, Computer Aided Systems Theory – EUROCAST 2007, Springer, 2007.
[4] Carnegie Mellon University. Alice Education Software, available at http://www.cmu.edu/corporate/news/2007/features/alice.shtml
[5] K.E. Wiegers. Peer Reviews in Software,Addison Wesley, London, UK, 2002.
[6] A. Spillner, T. Linz, H. Schaefer. Software Testing Foundations, dpunkt, 2006.
[7] Rohaida Romli, Shahida Sulaiman, Kamal Zuhairi Zamli, Automatic Programming Assessment and Test Data Generation-A review on its approaches, 2010 IEEE
[8] Tho T. Quan, Phung H. Nguyen, Thang H. Bui, Linh V. Huynh and Anh T. Do, A Framework For Automatic Verification Of Programing Exercises. 2009 IEEE
[9] Yingli Liang, Quanbo Liu, Jun Xu, Dongqing Wang, The Recent Development of Automated Programming Assessment, 2009 IEEE
[10] Khirulnizam Abd Rahman, Md Jan Nordin and Che Wan Shmsul Bahri C. W. A. Automated Programming Assessment using the Pseudocode Comparison Technique: Does It Really Work? IEEE 2008
[11] C. A. R. Hoare. An axiomatic basis for computer programming Communications of the ACM, 12(10):576–580,583, 1969.

Authors

Surendra Gupta received the Bachelor of Engineering degree in Computer Science and Engineering from Barktuulha University in 1997 and master of engineering de gree in computer engineering from DAVV University in 2000. He is currently working as Assistant Professor in SGSITS Indore in Computer Engineering Department. His interests are in empirical software engineering and software qualities. He is a member of the computer society of India.

Shiv Kumar Dubey received the bachelor of Technology degree in Information Technology from Mahatma Gandhi Chitrakoot Gramoday Vishwavidyalaya, Satna, Ma dhya Pradesh in 2008 and Master of Engineering degree in computer engineering from RGPV Bhopal MP in 2011.