An Educational System that Explains the Domain-oriented-explanation of Program’s Behaviors

Fumihiko Anma1, Tomonori Kato2, Tatsuhiro Konishi2, Yukihiro Itoh2 and Toshio Okamoto1
1Graduate School of Information Science, The University of Electro-Communications,
2Faculty of Information Shizuoka University
1{anma, okamoto}@ai.is.uec.ac.jp, 2{cs0025, konishi, itoh}@inf.shizuoka.ac.jp

Abstract

We have developed an educational system that helps novice programming language learners. Our system can output the verbal and visual explanation that explains the behavior of a program on the domain world. When learners input their own program that has a few bugs, they can spontaneously notice that there are a few bugs in their program by referring the explanation contradictory to their expectations, and try to fix them.

1. Introduction

In this paper, we introduce an educational system that helps novice programming language learners by explaining domain oriented functions of programs and its experimental evaluation. Our system analyzes a program on the basis of the domain world model in order to extract its domain oriented function and explains the function to learners.

Many systems supporting programming learners have been already proposed and used. Zeus[1], Tango[2] can generate visual explanations of high quality by using concrete objects on the domain world. Comparing these systems, our system can generate a visual explanation without embodying any special commands to programs. It generates explanations on the basis of the result of ‘simulation based program understanding’. It can accept even buggy programs to generate verbal and visual explanations of buggy behaviors of the programs. If learners input their own programs with some bugs, the learners can realize the existence of bugs and try to fix them by referring the explanations of the system.

2. The Programming Learning System

2.1. The Outline of the System

We show the configuration of our system in Figure.1. The static analyzer parses target programs, and obtains information about data flow, relations between array variables and their indexes. The simulation based program analyzer analyzes behaviors and observes useful characteristics. The explanation generator generates both explanations.

2.2 The Simulation Based Program Understanding

In order to extract behaviors of the program on the data structure world and the domain world, our system has models of both worlds. The system simulates the program on both models. The behavior of the program is gotten as a sequence of states of the domain model. The correspondence between individual behaviors and statements of the program are extracted from histories of simulations. In order to recognize each domain oriented function of each part of the program, the system has mechanisms called ‘observers’. Here, we can regard domain oriented functions as ones that make typical and useful characteristics on the domain appear. We have constructed two world models, the ‘greater or lesser world model’ for the exercises that can be solved by paying attention to numerical order and the ‘two dimensional space world model’ that is used for some numerical analyzing methods.

2.3 Verbal Explanations

The system outputs a hierarchical explanation. The hierarchy is constructed based on the structure of a
target program. First, a summarized explanation of the function of the program is shown to learners. More detailed explanation that explains the process of the function is shown to them only when they demand it. A result of an observation is translated into an explanation by applying the templates. For example, if the observer observes the maximum elements in an array in after state, the explanation generator selects the following template.

"Focus on [A] of [B]. Let's call this as [C][D]."

[A]: An object that is found by the observer to search a maximum/minimum number.
[B]: Objects that are observed by the observer in order to search the maximum/minimum value.
[C]: The class of an object [A]
[D]: The name of variable referring the object [A]

By using this template, the system generates an explanation such as "Focus on the greatest ball of the first-fourth balls. Let's call this as the ball max."

2.4 Visual Explanations

The procedure for visualization is as follows(Figure2).

Step0: The virtual screen to describe concepts is prepared. This screen is called as ‘drawing data’.
Step1: An initialization unit sets an initial state of the domain world.
Step2: The system simulates the behavior of a statement in the simulator. As a result, the change of the state caused by the behavior of the statement is observed by the observer.
Step3: The generating engine of the drawing data generates the each drawing data of the state of the domain world after executing each statement. The system recognizes the effect of the statement by the changes observed in the step2. For each type of effects, a function called the 'functions of generating drawing data' is prepared. Each function performs some operations such as addition, deletion, and movement of entities in order to update the drawing data according to the effect.
Step4: The figures that are equivalent to concepts included in the drawing data are drawn to the visualization window on graphical user interface.

3. An Example of our system

For example, assume the following situation: a learner tries to solve an exercise such as ‘Find the maximum number in the array.’ He thinks as following, and writes a buggy program shown in Figure4. Figure3 shows the output of our system.

It indicates explanation of behavior of the program on its domain. He will expect the program finds the maximum number, but the explanation does not tell such a function. Checking the process of action 2, he will confirm that each comparison is carried out as he intended. Then he will notice that his algorithm does not find the maximum number. In this case, the explanations help the learner to notice that the comparisons are executed independently and a result of each comparison doesn’t influence that next comparison.

4. Conclusion

In this paper, we described the outline of an educational system that helps novice programming language learners.

5. References