Jens Knoop, Uwe Zdun (Hrsg.): Software Engineering 2016,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2016 49

# Feedback Generation for Performance Problems in Introductory Programming Assignments

Sumit Gulwani[1] Ivan Radiček and Florian Zuleger[2]

**Abstract:** Providing feedback on programming assignments manually is a tedious, error prone, and time-consuming task. In [GRZ14], we motivate and address the problem of generating feedback on performance aspects in introductory programming assignments. We studied a large number of functionally correct student solutions to introductory programming assignments and observed: (1) There are different algorithmic strategies, with varying levels of efficiency, for solving a given problem. These different strategies merit different feedback. (2) The same algorithmic strategy can be implemented in countless different ways, which are not relevant for reporting feedback on the student program.

We propose a light-weight programming language extension that allows a teacher to define an algorithmic strategy by specifying certain key values that should occur during the execution of an implementation. We describe a dynamic analysis based approach to test whether a student's program matches a teacher's specification. Our experimental results illustrate the effectiveness of both our specification language and our dynamic analysis.

**Keywords:** Education, MOOCs, performance analysis, trace specification, dynamic analysis.

Providing feedback on programming assignments is a very tedious, error-prone, and time-consuming task for a human teacher, even in a standard classroom setting. With the rise of Massive Open Online Courses (MOOCs), which have a much larger number of students, this challenge is even more pressing. Hence, there is a need to introduce automation around this task. Immediate feedback generation through automation can also enable new pedagogical benefits such as allowing resubmission opportunity to students who submit imperfect solutions and providing immediate diagnosis on class performance to a teacher who can then adapt her instruction accordingly.

Recent research around automation of feedback generation for programming problems has focused on guiding students to functionally correct programs either by providing counterexamples (generated using test input generation tools) or generating repairs. However, non-functional aspects of a program, especially performance, are also important. We studied several programming sessions of students who submitted solutions to introductory C# programming problems on the PEX4FUN[3] platform. In such a programming session, a student submits a solution to a specified programming problem and receives a counterexample based feedback upon submitting a functionally incorrect attempt. The student may then inspect the counterexample and submit a revised attempt. This process is repeated

[1] Microsoft Research, USA
[2] TU Wien, Arbeitsbereich Formal Methods in Systems Design, Institut für Informationssysteme 184/4, Favoritenstraße 9–11, 1040 Wien, Austria
[3] http://www.pexforfun.com/

until the student submits a functionally correct attempt or gives up. We studied 24 different problems, and observed that of the 3993 different programming sessions, 3048 led to functionally correct solutions. However, unfortunately, on average around 60% of these functionally correct solutions had (different kinds of) performance problems. In this paper, we present a methodology for semi-automatically generating appropriate performance related feedback for such functionally correct solutions.

From our study, we made two observations that form the basis of our semi-automatic feedback generation methodology. (i) There are different *algorithmic strategies* with varying levels of efficiency, for solving a given problem. Algorithmic strategies capture the global high-level insight of a solution to a programming problem, while also defining key performance characteristics of the solution. Different strategies merit different feedback. (ii) The same algorithmic strategy can be implemented in countless different ways. These differences originate from local low-level implementation choices and are not relevant for reporting feedback on the student program.

In order to provide meaningful feedback to a student it is important to identify what algorithmic strategy was employed by the student program. A profiling based approach that measures running time of a program or use of static bound analysis techniques is not sufficient for our purpose, because different algorithmic strategies that necessitate different feedback may have the same computational complexity. Also, a simple pattern matching based approach is not sufficient because the same algorithmic strategy can have syntactically different implementations.

Our key insight is that the algorithmic strategy employed by a program can be identified by observing the values computed during the execution of the program. We allow the teacher to specify an algorithmic strategy by simply annotating (at the source code level) certain key values computed by a sample program (that implements the corresponding algorithm strategy) using a new language construct, called *observe*. Fortunately, the number of different algorithmic strategies for introductory programming problems is often small (at most 7 per problem in our experiments). These can be easily enumerated by the teacher in an iterative process by examining any student program that does not match any existing algorithmic strategy.

We propose a novel dynamic analysis that decides whether the student program (also referred to as an *implementation*) matches an algorithm strategy specified by the teacher in the form of an annotated program (also referred to as a *specification*). Our dynamic analysis executes a student's implementation and the teacher's specification to check whether the key values computed by the specification also occur in the corresponding traces generated from the implementation.

## References

[GRZ14] Gulwani, Sumit; Radicek, Ivan; Zuleger, Florian: Feedback generation for performance problems in introductory programming assignments. In: Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, (FSE-22), Hong Kong, China, November 16 - 22, 2014. pp. 41–51, 2014.