Open-Ended Tutoring for Programming: Building Next-Step Hints into an Online Development Environment

Kelly Rivers and Kenneth R. Koedinger

Carnegie Mellon University

Abstract. In prior work, we developed a new technique that uses solution spaces to generate hints for student automatically, without any need for instructor intervention [3]. Now, we have worked with our collaborators on the CloudCoder[2] project to integrate this hint system with their IDE, making it more accessible to students learning online. In this demo we will show the hints generated for students in multiple program states, describe how the algorithm adapts based on the amount of work a student has accomplished and how far they are from their goal, and demonstrate how new problems can be added to the system, both with and without prior sets of student data.

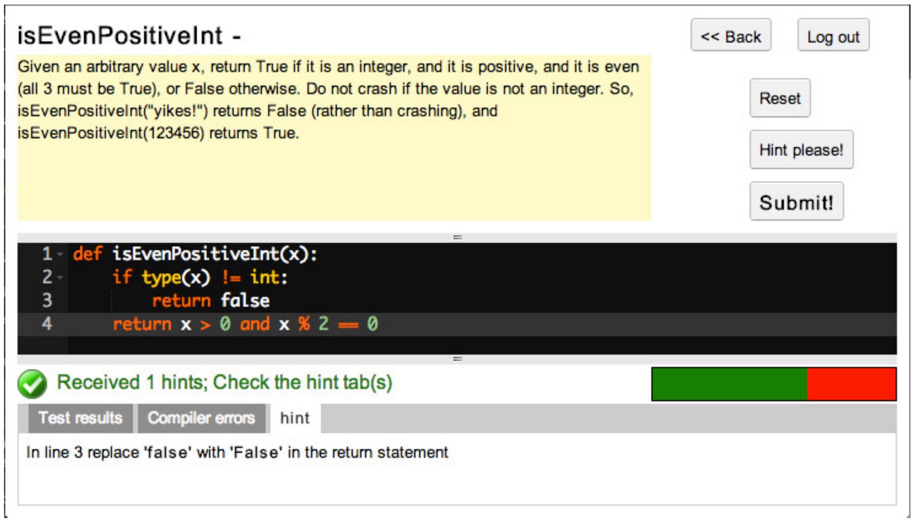
Keywords: automatic hint generation, solution space, programming tutors, open-ended tutors

1 Introduction

In any intelligent tutoring system, a student who is working on an individual problem needs to be able to access feedback and hints in order to progress independently in their work. In programming, feedback is usually accessible through automatic assessment [1], but hints are harder to generate when students are working on problems in an open-ended environment. Our recent research has shown that it is possible to use large corpuses of student data to generate hints automatically through the use of path construction algorithms [3]. We build on that research in the system presented here, with additional focus on how hint messages are formatted to align with the student's original intentions.

The system we describe is integrated into CloudCoder, an online IDE that lets teachers share problems across classes and assign work to their students online [2]. CloudCoder also provides students with compiler and test case feedback. We have extended this IDE by integrating a hint button which connects the CloudCoder server to our hint generation server, thereby allowing remote handling of hint messages (see Figure 1).

The hints generated by our system attempt to guide the student to the closest possible solution by indicating *where* they need to make a change, the *current* code that needs to be changed, and the *new* code that will take its place. In this


isEvenPositiveInt -

Given an arbitrary value x, return True if it is an integer, and it is positive, and it is even (all 3 must be True), or False otherwise. Do not crash if the value is not an integer. So, isEvenPositiveInt("yikes!") returns False (rather than crashing), and isEvenPositiveInt(123456) returns True.

```
def isEvenPositiveInt(x):
    if type(x) != int:
        return false
    return x > 0 and x % 2 == 0
```

Received 1 hints; Check the hint tab(s)

In line 3 replace 'false' with 'False' in the return statement

Fig. 1. An example of a hint produced for a student who has made a small error while programming, shown in CloudCoder.

demo we show a variety of hint messages that can be generated and explain how they are constructed within the system.

We will also demonstrate a few methods that can be used to create the solution space used to generate hints for the problems. These methods include gathering process data or final solutions from students, and also having the teacher generate multiple solutions ahead of time to pre-populate the solution space. We will give the members of the workshop the opportunity to generate solutions that can become part of a solution space for a new problem, and then test the resulting space to see how well it performs for a new incorrect solution.

Acknowledgements. This work was supported in part by Graduate Training Grant awarded to Carnegie Mellon University by the Department of Education (# R305B090023).

References

1. Douce, C., Livingstone, D., & Orwell, J. (2005). Automatic test-based assessment of programming: A review. Journal on Educational Resources in Computing (JERIC), 5(3), 4.
2. Hovemeyer, D., Hertz, M., Denny, P., Spacco, J., Papancea, A., Stamper, J., & Rivers, K. (2013, March). CloudCoder: building a community for creating, assigning, evaluating and sharing programming exercises. In Proceeding of the 44th ACM technical symposium on Computer science education (pp. 742-742). ACM.
3. Rivers, K., & Koedinger, K. R. (2013, June). Automatic generation of programming feedback: A data-driven approach. In The First Workshop on AI-supported Education for Computer Science (AIEDCS 2013) (p. 50).