Exploring the Design Space of Automatically Synthesized Hints for Introductory Programming Assignments

Ryo Suzuki [1], Gustavo Soares [2, 3], Elena Glassman [2], Andrew Head [2], Loris D’Antoni [4], and Bjørn Hartmann [2]


Background

For massive programming courses, personalized feedback does not scale.

Program synthesis techniques can enable personalized feedback at scale. It automatically find fix of students code and then turn this fix into a sequence of hints (e.g., AutoGrader [Singh 2013], Refazer [Rolim 2017]).

Problem

However, currently generated hints are different from teachers’ hint-giving strategy.

Findings from an Interview

We don’t want to give away the solution because it cuts off the learning opportunity. Students also do not like to have just an answer. (Interview with an intro CS course TA at UC Berkeley)

Challenges

A key challenge in automatic programming feedback is how to design pedagogically-useful hints as effective as manual teachers feedback.

Design Space of Synthesized Hints

1. location hints: point out locations that need to be fixed.
2. data hints: suggest the expected type or value of a variable at a code trace.
3. behavior hints: highlight how the incorrect behavior diverges from the nearest solution.
4. example hints: provide an example of input and output that a program must satisfy.
5. transformation hints: suggest abstract or concrete fixes to apply to incorrect code.

Contributions

1. a characterization of five types of hints that can be generated by state-of-the-art synthesis techniques, informed by a formative study
2. the implementation of these hints in an interactive debugging interface appropriate for deployment and evaluation in a large programming classroom.