# C S A I L MIT COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE LABORATORY

# OverCode: Visualizing Variation in Student Solutions to Programming Problems at Scale

Elena Glassman, Jeremy Scott, Rishabh Singh, Philip Guo, Rob Miller MIT CSAIL, University of Rochester Computer Science {elg,jks,rishabhs,rcm}@mit.edu, pg@cs.rochester.edu



## **Problem**

- In MOOCs, a single programming exercise may produce thousands of solutions from learners.
- •There may be several distinct, correct solutions to coding assignments.
- •Some solutions may be unknown to the teaching staff.
- •Understanding solution variation is important for providing appropriate feedback to students at scale.

# OverCode Approach

- •Visualize and explore thousands of programming solutions
- Use both static and dynamic analysis to cluster similar correct solutions
- •rename variables to create clean composite solutions that reflect student naming choices
- •scale linearly with solution length and number of solutions, unlike CodeWebs.1
- •Let instructors further filter and cluster solutions based on different criteria.

#### **Evaluation**

#### **Dataset of Solutions**

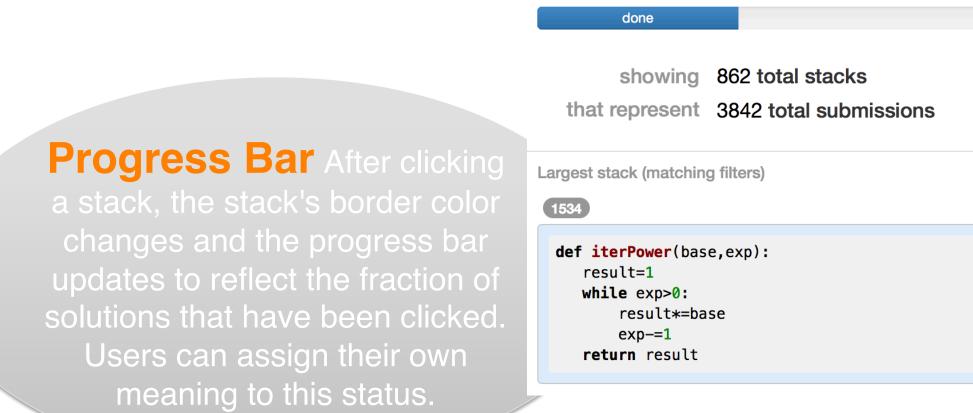
• We use a dataset of 1000's of *correct* solutions from 3 exercise problems in edX's 6.00x, an introductory programming course in Python.

#### **User Studies**

- We ran two user studies with 24 current and potential teaching assistants.
- Compared to an unclustered baseline, OverCode allowed instructors to
- more quickly develop a high-level view of students' understanding and misconceptions
- provide feedback that is relevant to more students.

## Summary

- An information visualization approach is necessary for instructors to explore the variations among solutions at the scale of MOOCs.
- OverCode is an important step towards that goal.



862 total stacks

## Filtering Solutions Tr

of lines of code by the number of solutions in which they appear. Clicking on a line of code adds it to the list of lines by which the stacks are filtered.

77 base=resultB

Filter

Rewrite-

lines that appear in at least 50

submissions

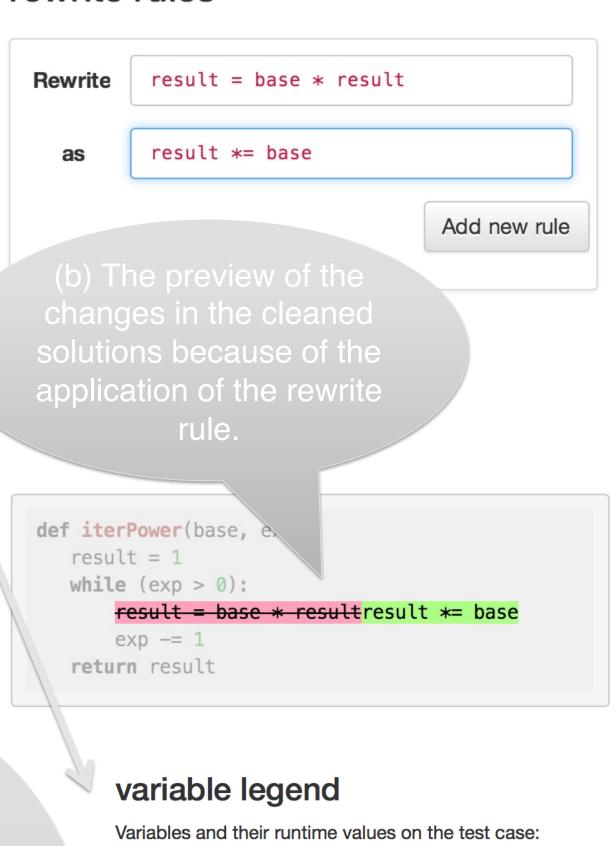
Legend

test case.

#### Rewrite Rules (a) An

example rewrite rule to replace all occurrences of statement result = base \* result with result \*= base.

#### rewrite rules



iterPower(5.0, 3)

**NUMBER** [3, 0, 1, 2]

**a** [5.0, 25.0]

base [5.0]

baseB [2, 5.0]

#### 3842 total 2592 def iterPower(base,exp): submissions 701 def iterPower(base,expB): Largest stack (matching filters) Remaining stacks (matching filters) 349 def iterPower(base,expC): 2<sup>nd</sup> largest id: 3 id: 51 def iterPower(base,expD): cluster... 51 def iterPower(resultB,expC): def iterPower(base,exp): def iterPower(base,exp): 55 elif expC==1: result=1 result=1 The largest 527 else: Variable while exp>0: while exp>0: cluster result=result\*base result\*=base 2466 exp-=1 Legends (stack) of exp=1exp=1solutions 279 exp=exp-1 return result return result shows the sequence 135 exp=expB of dynamic values 366 expC-=1 that all program **Shared Line Dimming** 3<sup>rd</sup> largest 65 expC=expC-1 variables in cleaned id: 727 Similar lines of code between 153 cluster... solutions take over 63 for i in rang two stacks are dimmed out the course of such that only differences def iterPower(base,exp): 174 for i in range execution on a given

OverCode

# between the two stacks are apparent. result=1 while exp>0: 52 iC+=1

filtering by:

nothing yet

- The top left panel shows the number of clusters, called *stacks*, and the total number of solutions visualized.
- The next panel down in the first column shows the largest stack, while the second column shows the remaining stacks.
- The third column shows the lines of code occurring in the cleaned solutions of the stacks together with their frequencies.

#### Acknowledgement

This material is based, in part, upon work supported by the National Science Foundation Graduate Research Fellowship (grant 1122374), the Microsoft Research Fellowship, the Bose Foundation Fellowship, and by Quanta Computer as part of the Qmulus Project. Any opinions, findings, conclusions, or recommendations in this paper are the authors', and do not necessarily reflect the views of the sponsors.

**b** [5.0, 4.0, 3.0, 2.0, 1.0, 0.0, 5.0, 4.0, 3.0, 2.0, 1.0, 0.0]

The OverCode user interface

iterPower