

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

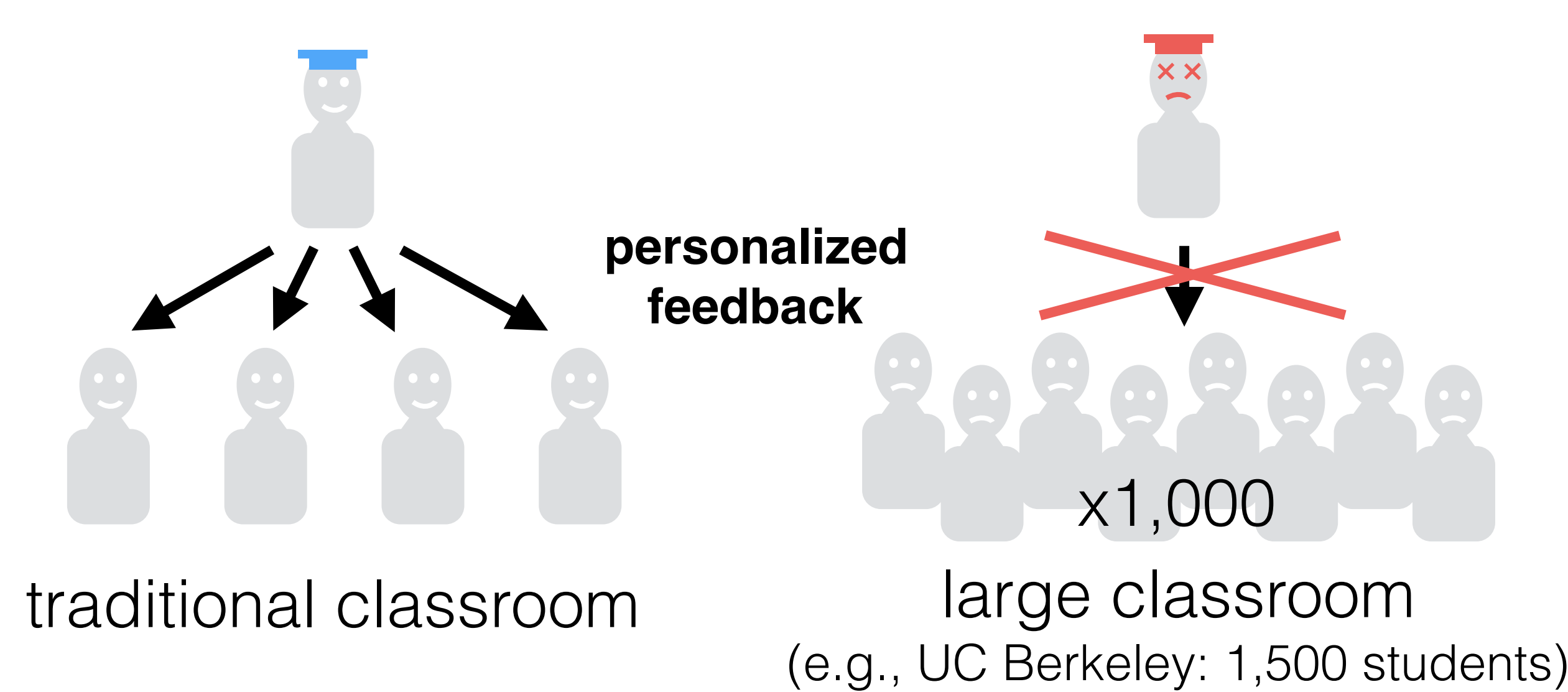
Ryo Suzuki <sup>[1]</sup>, Gustavo Soares <sup>[2, 3]</sup>, Elena Glassman <sup>[2]</sup>, Andrew Head <sup>[2]</sup>, Loris D'Antoni <sup>[4]</sup>, and Bjoern Hartmann <sup>[2]</sup>



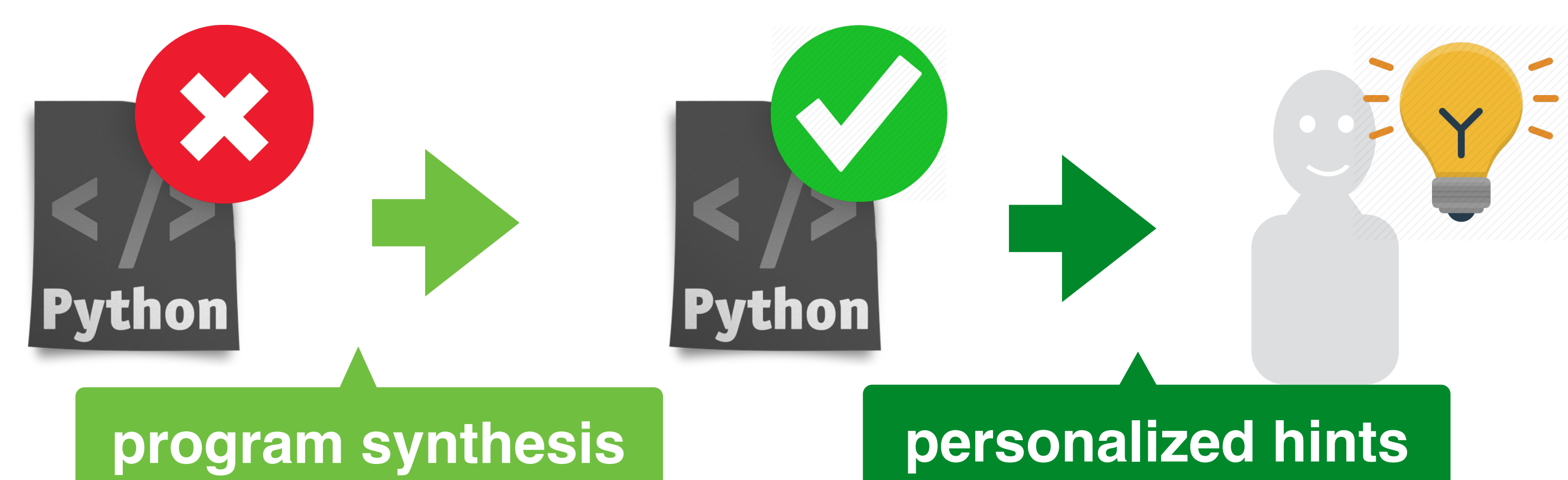
<sup>[1]</sup> University of Colorado Boulder, <sup>[2]</sup> UC Berkeley, <sup>[3]</sup> UFCG Brazil, and <sup>[4]</sup> University of Wisconsin-Madison

## 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])



**example**

```
def accumulate(combl
- total = 0
+ total = base
  if n==0:
    return combl
  else:
    while n>0:
      total :
```

**example**

Hint 1:  
Look at **line 2**

Hint 2:  
In total = 0 at line 2,  
**replace the value 0 with base.**

## Problem

However, **currently generated hints** are

e.g., at line 2 in "total = 0", replace the value 0 with base

**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)

### Observations of Q&A posts in Piazza



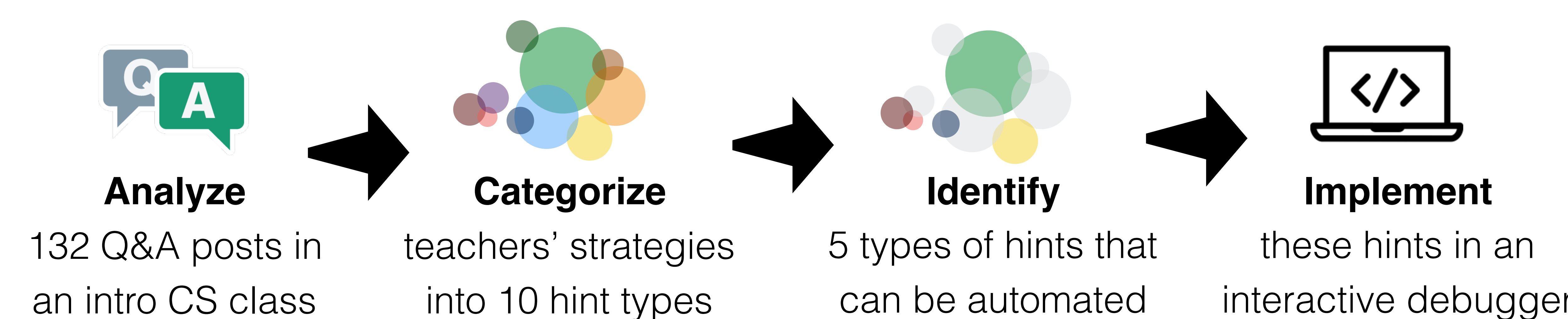
**Teachers illustrate why the code fails:** e.g.) Check the value of total and count. The correct behavior looks like this.  
total = 1 2 3 4 5 6 [7] 6 5  
count = 1 2 3 4 5 6 7 8 9

**Teachers encourage students to explore their code with interactive debuggers** e.g.) Try to examine the code in Python Tutor. What happens when you call accumulate? Is the combiner function that you're passing correct?

## 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



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

## Contributions

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