Writing Reusable Code Feedback at Scale with Mixed-Initiative Program Synthesis

Andrew Head*, Elena Glassman*, Gustavo Soares*, Ryo Suzuki, Lucas Figueredo, Loris D’Antoni, Björn Hartmann

* These three authors contributed equally to the work.
When Writing Feedback on Student Code, Teachers Can Draw on Deep Domain Knowledge

Incorrect Student Code Submissions

Teacher Comments

What happens when n is zero? Hint: look at lecture 5’s slides.

While this helper function is useful, it does not handle the case...

...but it does not scale.

Have you considered what would happen if combiner was set to?

Motivation
In lieu of Teacher-Written Feedback, Autograder Shows Test Cases

...but there’s still a gulf of evaluation.

Course Autograder
Program Synthesis Techniques Can Shrink the Gulf by Automatically Finding and Suggesting Bug Fixes for Students

**Student Submission**

```python
1 def product(n, term):
2     total, k = 0, 1
3      while k <= n:
4         total, k = total * term(k), k + 1
5     return total
```

**Test Case Results**

<table>
<thead>
<tr>
<th>Test</th>
<th>Input</th>
<th>Result</th>
<th>Expected</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(3, lambda x: x),</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(5, lambda x: x),</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>(3, lambda x: x * x)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>(5, lambda x: x * x)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In line 2, change total = 0 to total = 1

...but the automatically generated feedback is often mechanical, formulaic

Can we combine teachers’ deep domain knowledge with program synthesis to give students better feedback?
Learning Code Transformations from Pairs of Incorrect and Correct Submissions

Student 1 fixes iterative solution

Student 2 fixes recursive solution

Generalized code transformation

```
def product(n, term):
    total, k = 1, 1
    while k<=n:
        total = total*term(k)
        k = k+1
    return total
```

```
def product(n, term):
    if (n==1):
        return 1
    return product(n-1, term) * n
```

Insert

\[ \text{<exp>} \times \text{name} \rightarrow \text{<exp>} \times \text{term(name)} \]
Learning Bug-Fixing Code Transformations
We Scale Up a Little Teacher-Written Feedback by Attaching It to Code Transformations

Incorrect Student Code Submissions

<table>
<thead>
<tr>
<th>Submission 1</th>
<th>Submission 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="#" alt="Code" /></td>
<td><img src="#" alt="Code" /></td>
</tr>
</tbody>
</table>

Teacher Comments

What happens when \( n \) is zero?

Hint: look at lecture 5’s slides on base cases.

Motivation
Two Interfaces for Attaching Feedback to Code Transformations

MistakeBrowser: giving feedback on clusters

Learn transformations from Autograder

Collect feedback from teachers

Motivation

Related Systems: Divide and Conquer [ITS14], AutoStyle [ITS16]
Two Interfaces for Attaching Feedback to Code Transformations

**FixPropagator: attaching feedback to individual fixes**

Learns transformations from *and* collect feedback from...

Motivation
Our Program Synthesis Backend

**Refazer (ˈref.a.zɛ(r))/**
Means “To redo.”

Using Refazer [ICSE17] as a backend, our systems learn bug-fixing code transformations.
Contributions

• An approach for combining human expertise with program synthesis for delivering reusable, scalable code feedback

• Implementations of two different systems that use our approach: FixPropagator 🤖, MistakeBrowser 🔊

• In-lab studies that suggest that the systems fulfill our goals, also inform teachers about common student bugs
Outline

• Related Work
• Program Synthesis
• Systems
• Evaluation
System Design

Suggest fixes, feedback

Refazer
Program Synthesis [ICSE '17]

Interfaces for Teachers
[L@S '17]

Demonstrate fixes, write feedback

Mixed-initiative workflows
Teacher

Uploads test cases

Test 1
...
Test N

System

Learns transformations

Trans 1
...
Trans N

Clusters submissions by transformation

Finds transformation that fixes next submission

... and returns feedback written for it

Students

Submit code

incorrect submissions
final correct submission

Submits incorrect code

... Next Semester

Systems: MistakeBrowser
Return the product of the first $n$ terms in a sequence.

$n$ — a positive integer

term — a function that takes one argument

```python
>>> product(3, identity) # 1 * 2 * 3
6
>>> product(5, identity) # 1 * 2 * 3 * 4 * 5
120
>>> product(3, square)  # 1^2 * 2^2 * 3^2
36
>>> product(5, square)  # 1^2 * 2^2 * 3^2 * 4^2 * 5^2
14400
```

**Cluster 1**

- return term(n)*term(n-1)
- return term(n)*product(n-1, term)
Return the product of the first \( n \) terms in a sequence.

\( n \) -- a positive integer
\( \text{term} \) -- a function that takes one argument

```
>>> product(3, identity) # 1 * 2 * 3
6
>>> product(5, identity) # 1 * 2 * 3 * 4 * 5
120
>>> product(3, square) # 1^2 * 2^2 * 3^2
36
>>> product(5, square) # 1^2 * 2^2 * 3^2 * 4^2 * 5^2
14400
```
Looks like you're writing a recursive call. What might you be missing to enable recursion?
But Not All Classes Have Submission Histories for Hundreds of Students

Submit code

incorrect submissions
Student Submission

You can edit this code. ☑ Show original ☑ Edit ☑ Show diff

```python
1 def product(n, term):
2     return term(n) * product(n - 1, term)
3
4 Run tests again
```

Test results: Some tests failed

<table>
<thead>
<tr>
<th>Test</th>
<th>Input</th>
<th>Result</th>
<th>Expected</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(3, lambda x: x),</td>
<td>RecursionError</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(5, lambda x: x),</td>
<td>RecursionError</td>
<td>120</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>(3, lambda x: x * x),</td>
<td>RecursionError</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>(5, lambda x: x * x),</td>
<td>RecursionError</td>
<td>14400</td>
<td></td>
</tr>
</tbody>
</table>

Print output (test case 1)

```
RecursionError: ('maximum recursion depth exceeded',)
```

[This test case produced no console output.]
```python
1 def product(n, term):
2     if n == 0:
3         return 1
4     return term(n) * product(n - 1, term)
```

Test results: All tests succeeded

<table>
<thead>
<tr>
<th>Test</th>
<th>Input</th>
<th>Result</th>
<th>Expected</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(3, lambda x: x),</td>
<td>6</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(5, lambda x: x),</td>
<td>120</td>
<td>120</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>(3, lambda x: x * x),</td>
<td>36</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>(5, lambda x: x * x),</td>
<td>14400</td>
<td>14400</td>
<td></td>
</tr>
</tbody>
</table>

Print output (test case 1)

[This test case produced no console output.]
New Student Submission with Same Bug

Suggested Fix
Student Submission

You can edit this code.  ○ Show original  ○ Edit  ○ Show diff

1 def product(n, term):
  2   if n == 0:
  3     return 1
  4   if n != 0:
  5     return term(n) * product(n - 1, term)

Run tests again

Test results: All tests succeeded

<table>
<thead>
<tr>
<th>Test</th>
<th>Input</th>
<th>Result</th>
<th>Expected</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(3, lambda x: x),</td>
<td>6</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(5, lambda x: x),</td>
<td>120</td>
<td>120</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>(3, lambda x: x * x),</td>
<td>36</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>(5, lambda x: x * x),</td>
<td>14400</td>
<td>14400</td>
<td></td>
</tr>
</tbody>
</table>

Print output (test case 1)

[This test case produced no console output.]

Feedback

Student error detected.
This wrong answer can be fixed with the edits for submission 281.
This is the fix:

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>
| 1 | def product(n, term):
| 2 |   if n == 0:
| 3 |     return 1
| 4 |   if n != 0:
| 5 |     return term(n) * product(n - 1, term)

-- Apply this fix to the student's code

Another student with this same problem has already been given feedback. Do you want to use the feedback for them here?

-- Use existing feedback --

Notes

Submit feedback

Systems: FixPropagator
Both Fixes and Feedback Can Be Further Modified
# A Study of the Systems

**Participants:** Current and former teaching staff from CS1

- **MistakeBrowser** ($N = 9$)
- **FixPropagator** ($N = 8$)

<table>
<thead>
<tr>
<th>Interface Walkthrough</th>
<th>5 mins.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Main Task</strong></td>
<td>(30 mins.): Giving feedback on student submissions</td>
</tr>
<tr>
<td><strong>Measurements:</strong></td>
<td>Feedback, Manual corrections, Response to feedback recommendations (accepted, changed, rejected), Between-task surveys...</td>
</tr>
</tbody>
</table>

| **Qualitative Feedback:** | Survey and Post-interview |
1. Can a few manual corrections fix many submissions?
1. Can a few manual corrections fix many submissions?

FixPropagator propagates fixes from dozens of corrections to hundreds of submissions.
1. Can a few manual corrections fix many submissions?

**FixPropagator** propagates fixes from dozens of corrections to hundreds of submissions.

- Fixes were propagated within minutes ($median = 2m20s$, $\sigma = 7m34s$ for each correction).
2. How often is a teacher’s feedback relevant when it is matched to other students’ submission?
Feedback propagated with FixPropagator was correct a majority of the time, but not always.

Teachers reused feedback a median of 20 times, modifying it a median of 6 times (30%).

Generalizable Comment
“Check if you have the product of the correct number of terms.”

Non-Generalizable Comment
“Your starting value of z should be a function, not an int.”
2. How often is a teacher’s feedback relevant when it is matched to other students’ submission?

MistakeBrowser created conceptually consistent clusters of student bugs.
2. How often is a teacher’s feedback relevant when it is matched to other students’ submission?

MistakeBrowser created conceptually consistent clusters of student bugs.

![Bar chart showing the distribution of how often feedback is relevant when matched to other students' submissions.]

**Do these submissions share the same misconception?**

Responses for \( N = 11 \) clusters
Evaluation Questions

1. Can a few manual corrections fix many submissions?

   With a median of 10 corrections, FixPropagator suggested fixes for a median of 201 submissions.

2. How often is a teacher’s feedback relevant when it is matched to another student submission?

   Matched feedback was relevant ~75% of the time.
Limitations

• The impact of teacher feedback on student learning outcomes has not been evaluated.

• Code transformations were created that fix submissions one or two bugs away from correct.
Conclusion

We present an approach for combining human expertise with program synthesis for delivering reusable, scalable code feedback.

And two systems implementing this approach:

- MistakeBrowser
- FixPropagator
Conclusion

We present an approach for combining human expertise with program synthesis for delivering reusable, scalable code feedback.

And two systems implementing this approach:

MistakeBrowser   FixPropagator

Questions?