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.
Have you considered what would happen if combiner was set
incorrectly?

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.

When Writing Feedback on Student Code, Teachers Can Draw on Deep Domain Knowledge

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

...but there’s still a gulf of evaluation.
Program Synthesis Techniques Can Shrink the Gulf by Automatically Finding and Suggesting Bug Fixes for Students

Motivation

Student Submission

Test Case Results

...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

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

Student 2 fixes recursive solution

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

Generalized code transformation

<exp> * <name>  ➡️  <exp> * term(<name>)
Learning Bug-Fixing Code Transformations

Program Synthesis

Motivation
We Scale Up a Little Teacher-Written Feedback by Attaching It to Code Transformations

Incorrect Student Code Submissions

### Submission 1

```
1 1 def accumulate(combiner, base, n, term):
2 2   def prtti(combiner, n, term):
3 3     if n==1:
4 4       return term(n)
5 5       return combiner(term(n), prtti(combiner, n, term))
6 6 +   if n==0:
7 7 +     return base
```

Return combiner(base, prtti(combiner, n, term))

### Submission 2

```
1 1 def accumulate(combiner, base, n, term):
2 2   find_value = term(n)
3 3 +   if n==0:
4 4 +     return base
5 5 +     def find_value(combiner, base, n, term, value):
6 6       if n==1:
7 7       return combiner(base, value)
8 8     else:
9 9       return find_value(combiner, base, n-1, term, value)
```

Teacher Comments

**Code Transformation**

(add base case)

**What happens when n is zero?**

**Hint:** look at lecture 5’s slides on base cases.
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 (/hɛ.ʃa.'ze(h)/)**
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

[LS ’17]

Demonstrate fixes, write feedback

Mixed-initiative workflows
Teacher

- Uploads test cases
- Test 1
  - ... 
  - Test N

System

- Learns transformations
- Clusters submissions by transformation
  - Trans 1
  - ... 
  - Trans N

- 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
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

#### Examples of applied fix

- `return term(n)*term(n-1)`
- `return term(n)*product(n-1, term)`

### Submissions

#### Submission 1

```
1 1 def product(n, term):
2   if n == 1:
3     return 1
4   else:
5     return term(n)*term(n-1)
6     return term(n)*product(n-1, term)
```

#### Input

- `product(5, identity)`

#### Test feedback

<table>
<thead>
<tr>
<th>Input</th>
<th>Expected</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>product(5, identity)</code></td>
<td>120</td>
<td>20</td>
</tr>
</tbody>
</table>

#### Submission 2

```
1 1 def product(n, term):
2   total = 1
3   def b(n):
4     if n == 1:
5       return 1
6     def b(n-1):
7       return term(n)
8       return term(n)*product(n-1, term)
9       return b(n)*product(n-1, term)
```

#### Input

- `product(5, identity)`

#### Test feedback

<table>
<thead>
<tr>
<th>Input</th>
<th>Expected</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>product(5, identity)</code></td>
<td>120</td>
<td>20</td>
</tr>
</tbody>
</table>

#### Submission 3

```
1 1 def product(n, term):
2   if n == 1:
3     return 1
```

#### Hints

- Set Hint
- Reuse previous hints
Assignment description

Return the product of the first \( n \) terms in a sequence.

\( n \quad \text{-- a positive integer} \)

\( \text{term} \quad \text{-- a function that takes one argument} \)

```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

Cluster 1

Examples of applied `fn`

- return \( \text{term}(n) \cdot \text{term}(n-1) \)

- return \( \text{term}(n) \cdot \text{product}(n-1, \text{term}) \)

Submissions

Select all submissions

Submission 1

```python
1 1 def product(n, term):
2   if n == 1:
3     return 1
4   else:
5     return term(n) * term(n-1)

Test feedback

<table>
<thead>
<tr>
<th>Input</th>
<th>Expected</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>product(5, identity)</td>
<td>120</td>
<td>20</td>
</tr>
</tbody>
</table>
```

Submission 2

```python
1 1 def product(n, term):
2   total = 1
3   def a(n):
4     if n == 1:
5       return 1
6     def b(n):
7       return term(n)
8     return 2 * b(n) * product(n-1, term)
9     return a(n)

Test feedback

<table>
<thead>
<tr>
<th>Input</th>
<th>Expected</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>product(5, identity)</td>
<td>120</td>
<td>20</td>
</tr>
</tbody>
</table>
```

Submission 3

```python
1 1 def product(n, term):
2   if n == 1:
3     return 1
```

Systems: MistakeBrowser
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
Teacher

Uploads test cases

Test 1
... Test N

System

Fixes
Picks submission

Corrects

Writes hint

Fixes

Systems: MistakeBrowser

Systems: FixPropagator

Submits incorrect submissions

Submit code

Students

Returns feedback to students

Accepts or modifies suggested fixes, feedback

Learns transformations, makes clusters, attaches feedback

Suggests fixes and feedback

Systems: FixPropagator
**Student Submission**

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

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

---

**Feedback**

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.]
def product(n, term):
    if n == 0:
        return 1
    return term(n) * product(n - 1, term)
New Student Submission with Same Bug

Suggested Fix
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 \( N = 8 \)

**Interface Walkthrough** (5 mins.)

**Main Task** (30 mins.): Giving feedback on student submissions

**Measurements:** Feedback, Manual corrections, Response to feedback recommendations (accepted, changed, rejected), Between-task surveys...

**Qualitative Feedback:** Survey and Post-interview
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?
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%).

<table>
<thead>
<tr>
<th>Generalizable Comment</th>
<th>Non-Generalizable Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Check if you have the product of the correct number of terms.”</td>
<td>“Your starting value of z should be a function, not an int.”</td>
</tr>
</tbody>
</table>
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.

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?