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

Submission 1
```python
def accumulate(combiner, base, n, term):
    def fterm(combiner, n, term):
        if n==1:
            return term(n)
        return combiner(term(n), fterm(combiner, n-1, term))

    return combiner(base, fterm(combiner, n, term))
```

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

Submission 2
```python
def accumulate(combiner, base, n, term):
    value = term(n)
    def fterm(combiner, base, n, term, value):
        if n==1:
            return combiner(base, value)
        else:
            return combiner(find_value(combiner, base, n-1, term, comb), fterm(combiner, base, n, term, value))

    return fterm(combiner, base, n, term, value)
```

While this helps... but it does not scale.

Submission 3
```python
def accumulate(combiner, base, n, term):
    def fterm(combiner, n, term):
        if n==1:
            return term(n)
        return combiner(term(n), fterm(combiner, n-1, term))

    return combiner(base, fterm(combiner, n, term))
```

Have you considered what would happen if combiner was set...
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**

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
        k = k+1
    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
We Scale Up a Little Teacher-Written Feedback by Attaching It to Code Transformations

<table>
<thead>
<tr>
<th>Incorrect Student Code Submissions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Submission 1</strong></td>
</tr>
<tr>
<td>1 @ -1.6 +1.0 @</td>
</tr>
<tr>
<td>1 1 def accumulate(combiner, base, n, term):</td>
</tr>
<tr>
<td>2 2   def prttii(combiner, n, term):</td>
</tr>
<tr>
<td>3 3     if n=1:</td>
</tr>
<tr>
<td>4 4       return term(n)</td>
</tr>
<tr>
<td>5 5       return combiner(term(n), prttii(combiner, n))</td>
</tr>
<tr>
<td>6 6       if n==0:</td>
</tr>
<tr>
<td>7 7       return base</td>
</tr>
<tr>
<td>8 8       return combiner(base, prttii(combiner, n, term))</td>
</tr>
</tbody>
</table>

| **Submission 2**                  |
| 1 @ -1.8 +1.0 @                   |
| 1 1 def accumulate(combiner, base, n, term): |
| 2 2   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, value) |
| 10 10       return find_value(combiner, base, n, term, val) |

**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

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...

Teacher picks a submission → Teacher fixes submission and writes a hint

Feedback Bank
Our Program Synthesis Backend

**Refazer** (/rɛˌfa.ˈze(ə)/)
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
Related Work

Program Synthesis for Generating Feedback

```
def computeDeriv(poly):
    deriv = []
    zero = 0
    if (len(poly) == 1):
        return deriv
    for expo in range(0, len(poly)):
        if (poly[expo] == 0):
            zero += 1
        else:
            deriv.append(poly[expo])
    return deriv
```

(a) Student’s solution  
(b) Generated Feedback

The program requires 3 changes:
- In the return statement `return deriv` in line 5, replace `deriv` by `[0].`
- In the comparison expression `(poly[expo] == 0)` in line 7, change `(poly[expo] == 0)` to `False.`
- In the expression `range(0, len(poly))` in line 6, increment 0 by 1.

Figure 2. (a) A student’s computeDeriv solution from the 6.00x discussion board and (b) the feedback generated by our tool on this solution.

AutoGrader [PLDI13]

... and beyond CS1 assignments, AutomataTutor [TOCHI15], CodeAssist [FSE16], ...
Related Work

Interfaces for Giving Feedback on Submission Clusters

**Divide and Correct** [L@S14]

**AutoStyle** [ITS16]
Program Synthesis

Program synthesis can learn transformations *from demonstrations*.

<table>
<thead>
<tr>
<th>Full Name</th>
<th>Last Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andrew Head</td>
<td>Head</td>
</tr>
<tr>
<td>Elena Glassman</td>
<td>Glassman</td>
</tr>
<tr>
<td>Gustavo Soares</td>
<td>Soares</td>
</tr>
<tr>
<td>Ryo Sukuzi</td>
<td>Sukuzi</td>
</tr>
</tbody>
</table>

- Human demonstrates
- Synthesized program propagates
Learning Transformations from Demonstrations

Sources of demonstrations
- students debugging
- teachers correcting student code

Synthesized transformation:
Split string on space, return second substring

Synthesized transformation:
Before final return statement, insert AST node “if n==0: return base”
Propagating Transformations

<table>
<thead>
<tr>
<th>Name</th>
<th>Last</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andrew Head</td>
<td>Head</td>
</tr>
<tr>
<td>Elena Glassman</td>
<td>Glassman</td>
</tr>
<tr>
<td>Gustavo Soares</td>
<td>Soares</td>
</tr>
<tr>
<td>Ryo Sukuzi</td>
<td>Sukuzi</td>
</tr>
</tbody>
</table>

Synthesized program propagates

Program Synthesis
What Bug-Fixing Code Transformations Can Refazer Learn?

```
def accumulate(combiner, base, n, term):
    if n==1:
        return term(n)
    return combiner(term(n), accumulate(combiner, term(n), n-1, term))
```

Missing base cases...

```
def repeated(f, n):
    if n==0:
        return f
    return identity

else:
    return compose1(f, repeated(f, n-1))
```

Function substitutions… and so on…
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

System: MistakeBrowser

Submits incorrect code

Students

Submit code

Submits incorrect submissions

... Next Semester
## Assignment description

Return the product of the first n terms in a sequence.

- \( n \) — a positive integer
- \( \text{term} \) — 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
```

## Submissions

### Submission 1

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>def product(n, term):</td>
</tr>
<tr>
<td>2</td>
<td>if ( n = 1 ):</td>
</tr>
<tr>
<td>3</td>
<td>return 1</td>
</tr>
<tr>
<td>4</td>
<td>else:</td>
</tr>
<tr>
<td>5</td>
<td>return term(n)*term(n-1)</td>
</tr>
<tr>
<td>6</td>
<td>+</td>
</tr>
<tr>
<td>7</td>
<td>return term(n)*product(n-1, term)</td>
</tr>
</tbody>
</table>

### 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

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>def product(n, term):</td>
</tr>
<tr>
<td>2</td>
<td>total = 1</td>
</tr>
<tr>
<td>3</td>
<td>def a(n):</td>
</tr>
<tr>
<td>4</td>
<td>if ( n = 1 ):</td>
</tr>
<tr>
<td>5</td>
<td>return 1</td>
</tr>
<tr>
<td>6</td>
<td>else:</td>
</tr>
<tr>
<td>7</td>
<td>def b(n):</td>
</tr>
<tr>
<td>8</td>
<td>return term(n)</td>
</tr>
<tr>
<td>9</td>
<td>+</td>
</tr>
<tr>
<td>10</td>
<td>return b(n)*product(n-1, term)</td>
</tr>
</tbody>
</table>

### 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

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>def product(n, term):</td>
</tr>
<tr>
<td>2</td>
<td>if ( n = 1 ):</td>
</tr>
<tr>
<td>3</td>
<td>return 1</td>
</tr>
</tbody>
</table>

## Cluster 1

### Examples of applied fix

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

## Systems: MistakeBrowser
It 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

Teacher picks a submission and fixes it, writing a hint.

System learns transformations, makes clusters, and attaches feedback.

System suggests fixes and feedback.

Teacher accepts or modifies the suggested fixes and feedback.

Returns feedback to students.

Incorrect submissions are indicated.

Students submit code.
- x
- ...incorrect submissions

System picks a submission and suggests fixes and feedback.

System learns transformations, makes clusters, and attaches feedback.

Accepts or modifies the suggested fixes and feedback.
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)
```

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><img src="image" alt="Result" /></td>
</tr>
<tr>
<td>2</td>
<td>(5, lambda x: x),</td>
<td>RecursionError</td>
<td>120</td>
<td><img src="image" alt="Result" /></td>
</tr>
<tr>
<td>3</td>
<td>(3, lambda x: x * x),</td>
<td>RecursionError</td>
<td>36</td>
<td><img src="image" alt="Result" /></td>
</tr>
<tr>
<td>4</td>
<td>(5, lambda x: x * x),</td>
<td>RecursionError</td>
<td>14400</td>
<td><img src="image" alt="Result" /></td>
</tr>
</tbody>
</table>

Print output (test case 1)

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

[This test case produced no console output.]
```python
def product(n, term):
    if n == 0:
        return 1
    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
def product(n, term):
    if n == 0:
        return 1
    if n != 0:
        return term(n) * product(n - 1, term)
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?
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
  \((median = 2\text{m}20\text{s}, \sigma = 7\text{m}34\text{s} \text{ for each correction})\).
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.
Clusters Helped Teachers Give Feedback

Participants reported that the interfaces “gave me insight into student mistakes and misconceptions” ($\mu = 6.2$, $\sigma = 0.44$, range =1-7).

Seeing all of the similar instances of the same (or nearly the same) misconception was very useful, because it suggested ways to address common issues shared by many students.

- Participant 3, about MistakeBrowser
1. Can a few manual corrections fix many submissions?

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

Teacher fixes student submission and gives feedback

Time (minutes)
1. Can a few manual corrections fix many submissions?

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

Teacher fixes student submission and gives feedback

FixPropagator matches feedback to another submission

Time (minutes)
FixPropagator propagates fixes from dozens of corrections to hundreds of submissions.

1. Can a few manual corrections fix many submissions?
2. How often is a teacher’s feedback relevant when it is matched to other students’ submission?

Fixes that FixPropagator learned were typically correct when applied to other submissions.
2. How often is a teacher’s feedback relevant when it is matched to other students’ submission?

Fixes that FixPropagator learned were typically correct when applied to other submissions.
2. How often is a teacher’s feedback relevant when it is matched to other students’ submission?

Fixes that FixPropagator learned were typically correct when applied to other submissions.

"Propagating" a fix
 Fixes that FixPropagator learned were typically correct when applied to other submissions.

“Propagating” a fix
Fixes that FixPropagator learned were typically correct when applied to other submissions.

2. How often is a teacher’s feedback relevant when it is matched to other students’ submission?
Fixes that FixPropagator learned were typically correct when applied to other submissions.

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

Teachers applied a median of 20 fixes, and only modified those fixes a median of 3 times.
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.
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.
Evaluation

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.

Feedback “available”  “Reusing” feedback
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.
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.
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.
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.
2. How often is a teacher’s feedback relevant when it is matched to another student submission?

Teachers provided one piece of feedback on clusters that were mostly internally consistent.
FixPropagator propagates fixes from dozens of manual corrections to dozens of solutions.

![Graph showing the propagation of fixes over time.](image)

- **Generated feedback reused**
- **Generated feedback modified**
- **Generated feedback ignored**
- **Feedback written from scratch**
- **Feedback transferred to a new submission (each shade of gray is caused by a different fix)**

*Time (min)*
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?