

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

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

* These three authors contributed equally to the work.



Berkeley
UNIVERSITY OF CALIFORNIA



UFCG
Universidade Federal
de Campina Grande
SUPRA OMNES LUX LUCIS



University of Colorado
Boulder



WISCONSIN
UNIVERSITY OF WISCONSIN-MADISON

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

Incorrect Student Code Submissions

Submission 1 X

```
@@ -1,6 +1,8 @@
def accumulate(combiner, base, n, term):
    def prtii(combiner, n, term):
        if n==1:
            return term(n)
        return combiner(term(n), prtii(combiner, n-1, term))
    return combiner(base, prtii(combiner, n, term))
```

Submission 2 X

```
@@ -1,8 +1,10 @@
def accumulate(combiner, base, n, term):
    value = term(n)
    def find_value(combiner, base, n, term, value):
        if n==1:
            return combiner(base, value)
        else:
            return find_value(combiner, base, n-1, term, combi
return find_value(combiner, base, n, term, value)
```

Submission 3 X

```
@@ -1,7 +1,9 @@
def accumulate(combiner, base, n, term):
```

Teacher Comments

What happens when n is zero? Hint: look at lecture 5's slide

While this he

...but it does not scale.

Have you considered what would happen if `combiner` was so

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

Student Submission

A screenshot of a code editor interface. On the left, there is a vertical stack of tabs, with the top one labeled "Untitled-1". The main area contains Python code for calculating the product of the first n terms of a sequence. A "Run tests again" button is visible below the code. To the right, a "Test results" section shows four test cases with their inputs, expected outputs, and actual results. All four tests have passed.

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

Run tests again

Test results: All tests succeeded

Test	Input	Result	Expected	Output
1	(3, lambda x: x),	→ 6	6	💻
2	(5, lambda x: x),	→ 120	120	💻
3	(3, lambda x: x * x),	→ 36	36	💻
4	(5, lambda x: x * x),	→ 14400	14400	💻

Test Case Results

...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

A screenshot of a programming environment showing a student's code for a `product` function. The code has a syntax error in line 2 where `total` is assigned a value of `0, 1`. A red annotation with the number `1` highlights this line. Below the code is a button labeled `Run tests again`. A red message at the bottom of the interface says `In line 2, change total = 0 to total = 1`. To the left, under the heading `Test Case Results`, there is a table showing four test cases with their inputs and expected outputs.

Test	Input	Result	Expected	Output
1	(3, lambda x: x),			
2	(5, lambda x: x),			
3	(3, lambda x: x * x)			
4	(5, lambda x: x * x)			

...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

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

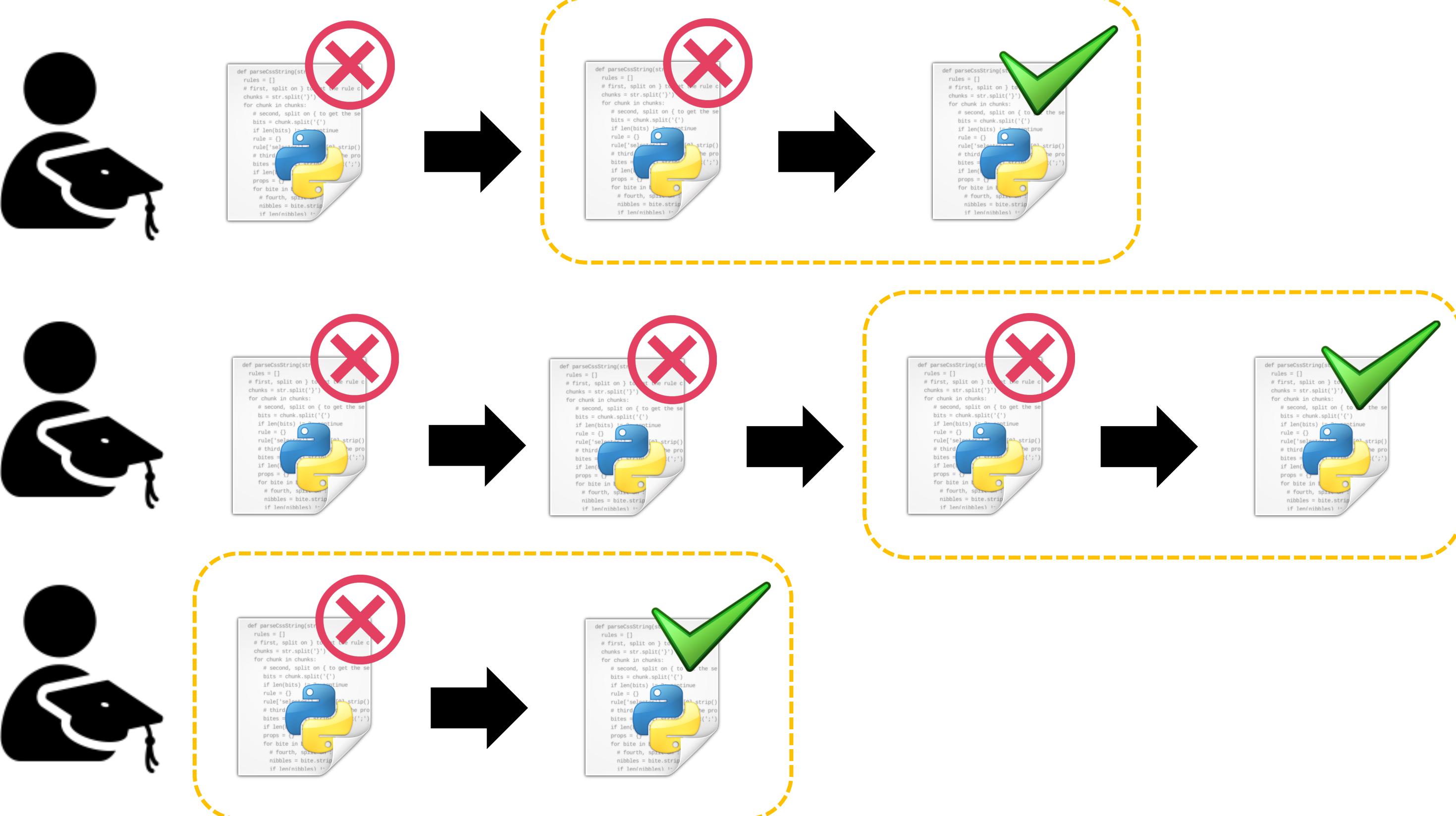
Student 2 fixes
recursive solution

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

Generalized code
transformation

$\langle \text{exp} \rangle * \langle \text{name} \rangle$  $\langle \text{exp} \rangle * \text{term}(\langle \text{name} \rangle)$

Learning Bug-Fixing Code Transformations



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

Incorrect Student Code Submissions

Submission 1 X

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

Submission 2 X

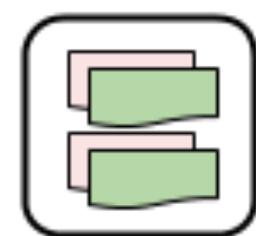
```
@@ -1,8 +1,10 @@
 1   1 def accumulate(combiner, base, n, term):
 2   2     value = term(n)
 3 + 3     if n==0:
 4 + 4     return base
 5   5     def find_value(combiner, base, n, term, value):
 6   6         if n==1:
 7   7             return combiner(base, value)
 8   8         else:
 9   9             return find_value(combiner, base, n-1,
10  10    return find_value(combiner, base, n, term, val
```

Code Transformation
(add base case)

Teacher Comments

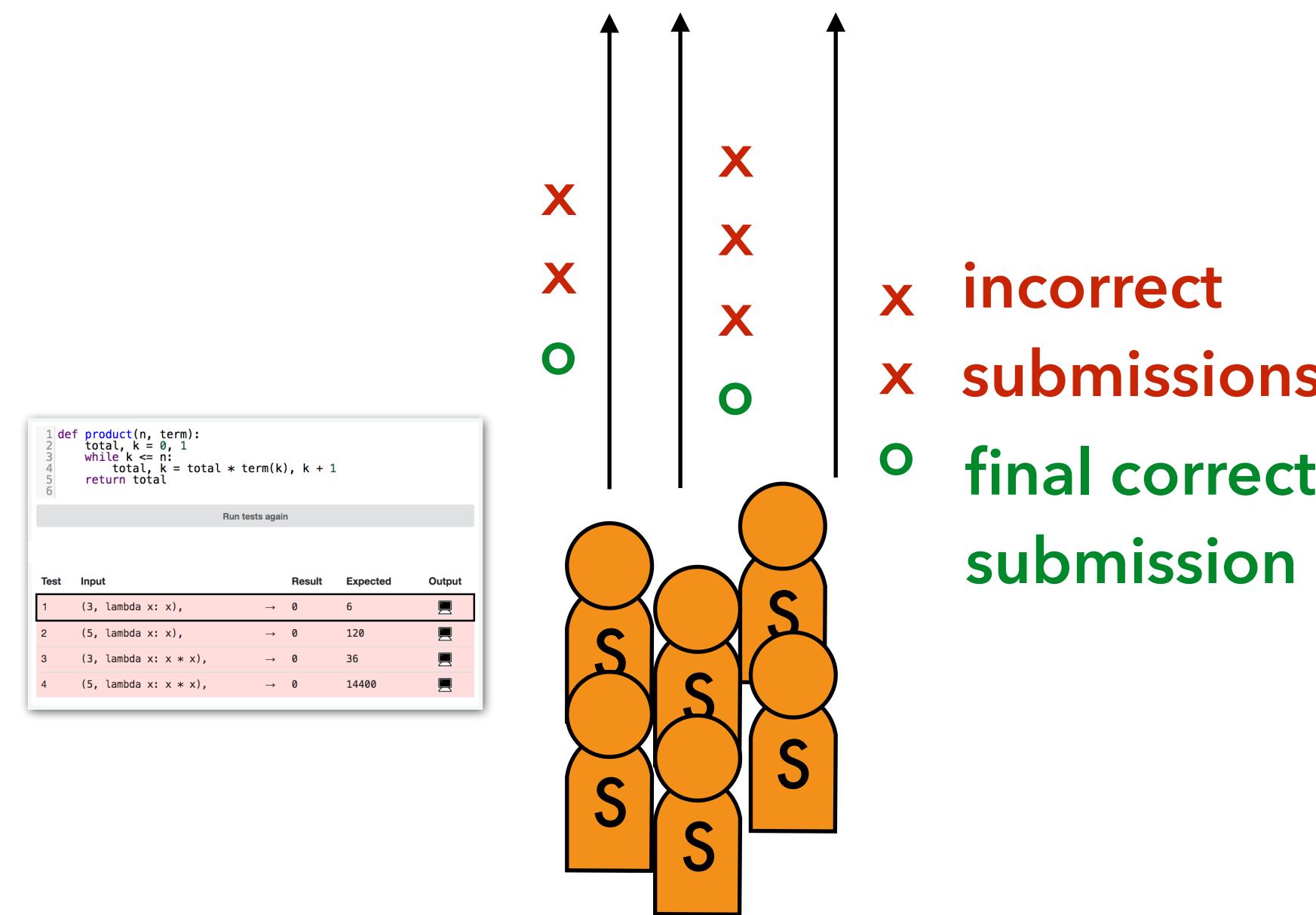
What happens when
n is zero?
Hint: look at lecture 5's
slides on base cases.

Two Interfaces for Attaching Feedback to Code Transformations

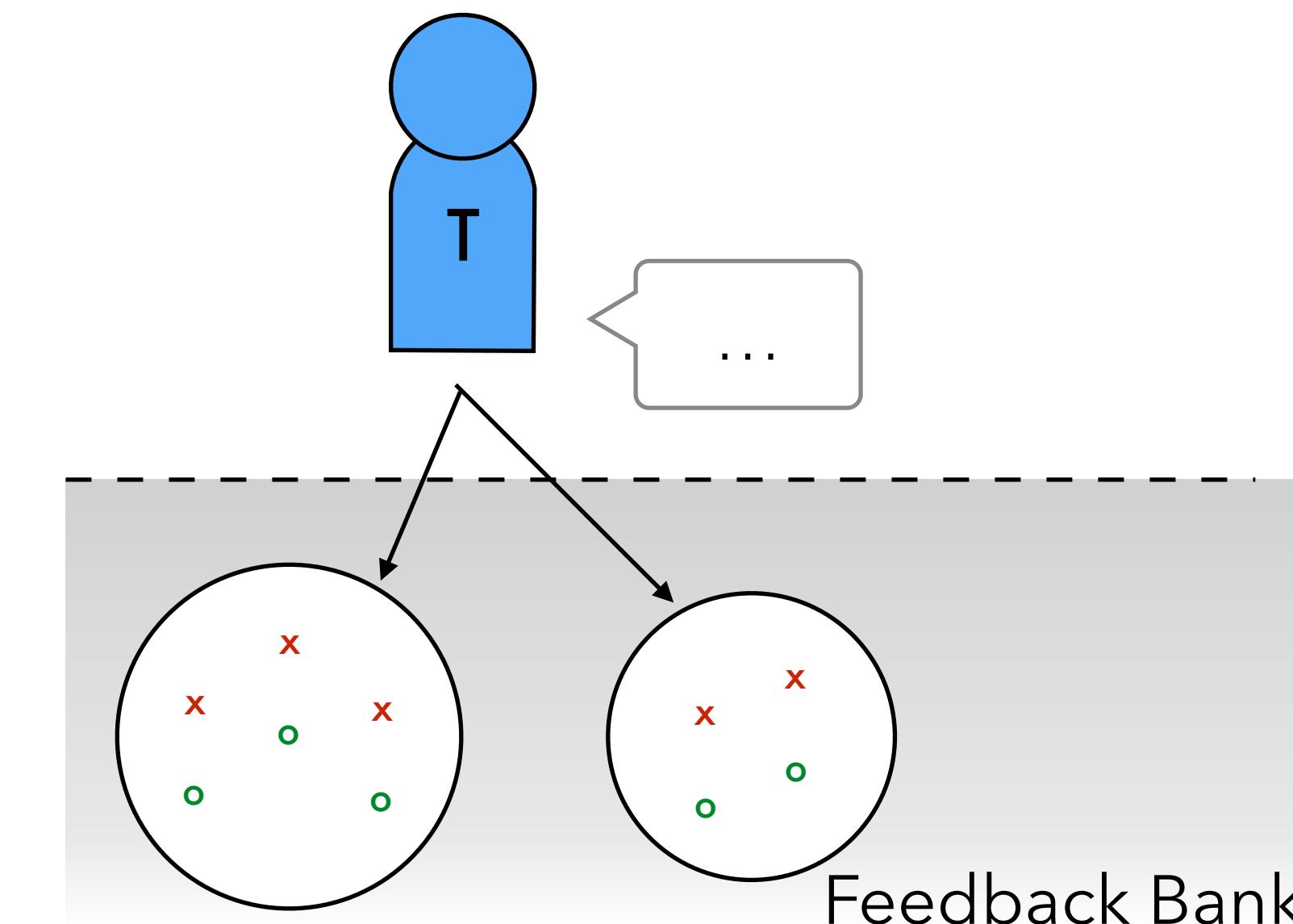


MistakeBrowser: giving feedback on clusters

Learn transformations from Autograder

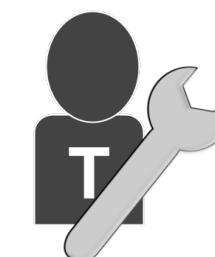


Collect feedback from teachers



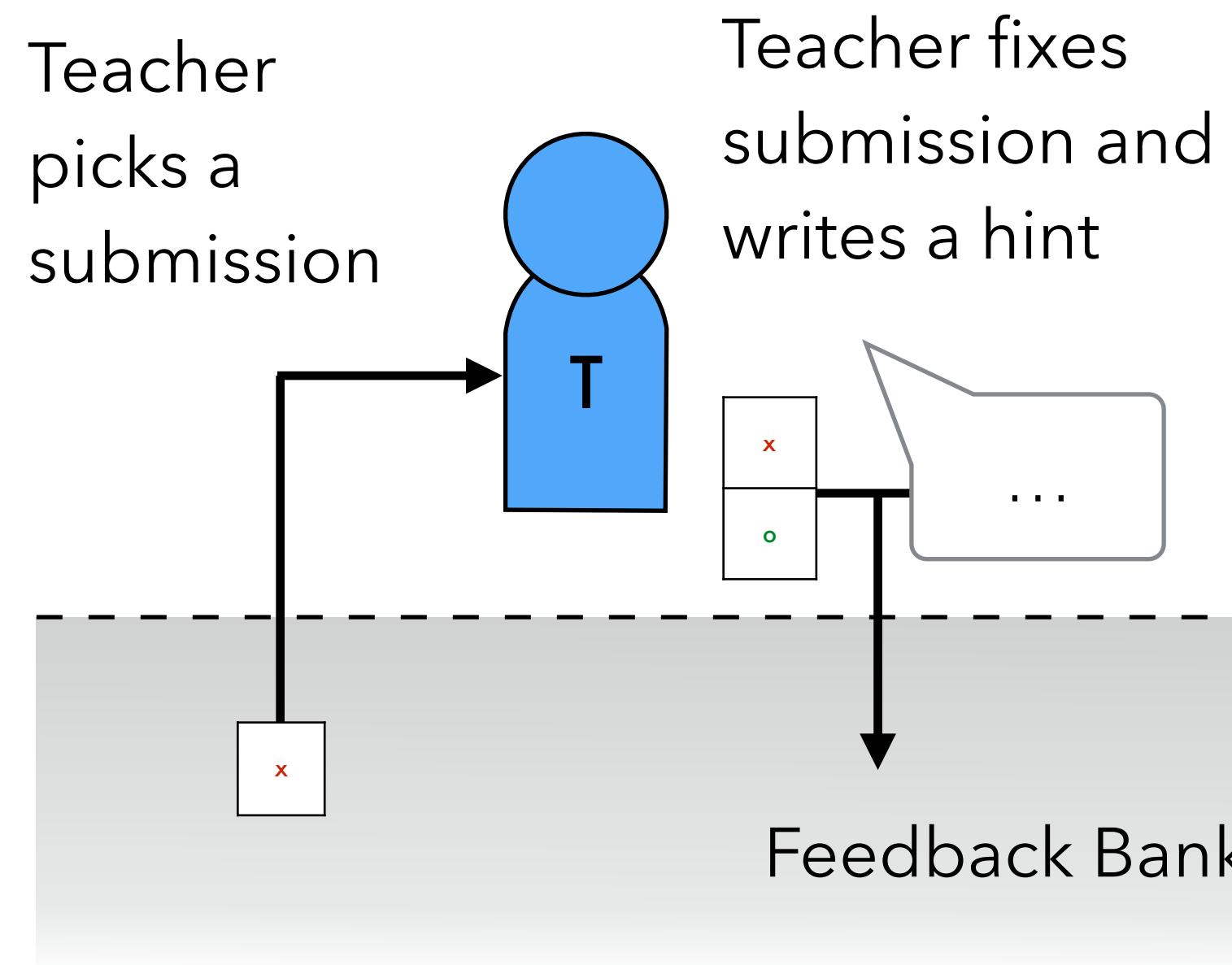
Related Systems: *Divide and Conquer* [ITS14], *AutoStyle* [ITS16]

Two Interfaces for Attaching Feedback to Code Transformations



FixPropagator: attaching feedback to individual fixes

Learns transformations from and collect feedback from...



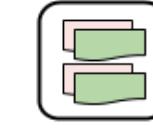
Our Program Synthesis Backend

Refazer (/hɛ.fə.'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**

Related Work

Program Synthesis for Generating Feedback

```
1 def computeDeriv(poly):
2     deriv = []
3     zero = 0
4     if (len(poly) == 1):
5         return deriv
6     for expo in range (0, len(po
7         if (poly[expo] == 0):
8             zero += 1
9         else:
10            deriv.append(poly[ex
11 return deriv
```

(a) Student's solution

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**.

(b) Generated Feedback

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

The screenshot shows the Power Grader interface for a question: "Who or what makes federal (national) laws in the US?". The interface is divided into several sections:

- Clusters:** A list of submissions grouped into clusters. One cluster, labeled **A**, contains the answer "legislative branch". Another cluster, labeled **B**, contains "senate and house of representatives".
- Subclusters:** A detailed view of cluster **A** showing individual submissions like "x103 legislative branch".
- Answers:** A list of student answers. One answer, labeled **D**, is "senate". Another answer, labeled **E**, is "the senate".
- Grading:** A panel for selecting a grade (Mark Unread, Grading, Incorrect, Partial, Correct) for the selected subcluster.
- Feedback:** A text input field for providing feedback, with the placeholder "the house also makes laws".
- Answer Key:** A section containing the correct answer "Congress" and other options like "Senate and House" and "Senate and House of Representatives".

The screenshot shows the AutoStyle interface for a Python challenge:

```
def num_common_letters(goal_word, guess):  
    goal=str(goal_word)  
    guess=str(guess)  
    n,i=0,0  
    while i<=len(goal)-1:  
        j=0  
        while j<=len(guess)-1:  
            if goal[i]==guess[j]:  
                n+=1  
                j=len(guess)  
            else:  
                j+=1  
        i+=1  
    return n
```

Charlie's solution

Here's some approach advice...

For each letter in the first word, think about what Python construct you can use to find whether it is also in the second word. You should not have to iterate over the second word. Use the skeleton we've provided to help you structure your solution.

Here's a skeleton of what your improved solution might look like...

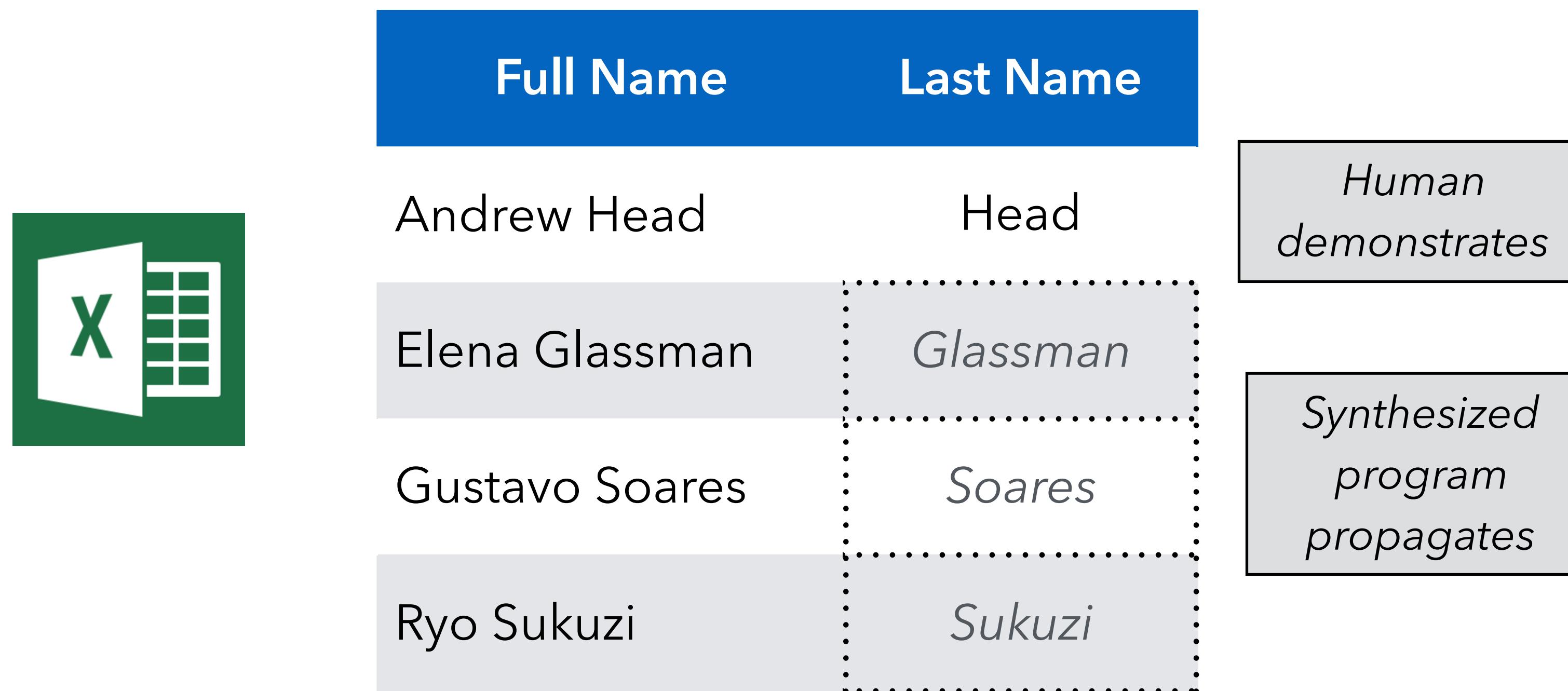
```
def num_common_letters(-----, -----):  
    ----- = -----  
    ----- = []  
    for - in list(-----):  
        if - in list(-----) ----- in -----:  
            ----- += -  
            ----- += -  
    return -----
```

Divide and Correct [L@S14]

AutoStyle [ITS16]

Program Synthesis

Program synthesis can learn transformations *from demonstrations*.



Learning Transformations from Demonstrations

Full Name Last Name

T

Sources of demonstrations

- students debugging
- teachers correcting student code

Ryo Sukuzi

Synthesized transformation:
Split string on space, return second substring

Submission 1 

```
@@ -1,6 +1,8 @@
def accumulate(combiner, base, n, term):
    def prtii(combiner, n, term):
        if n==1:
            return term(n)
        return combiner(term(n), prtii(combiner, n-1, term))
    return combiner(base, prtii(combiner, n, term))
```

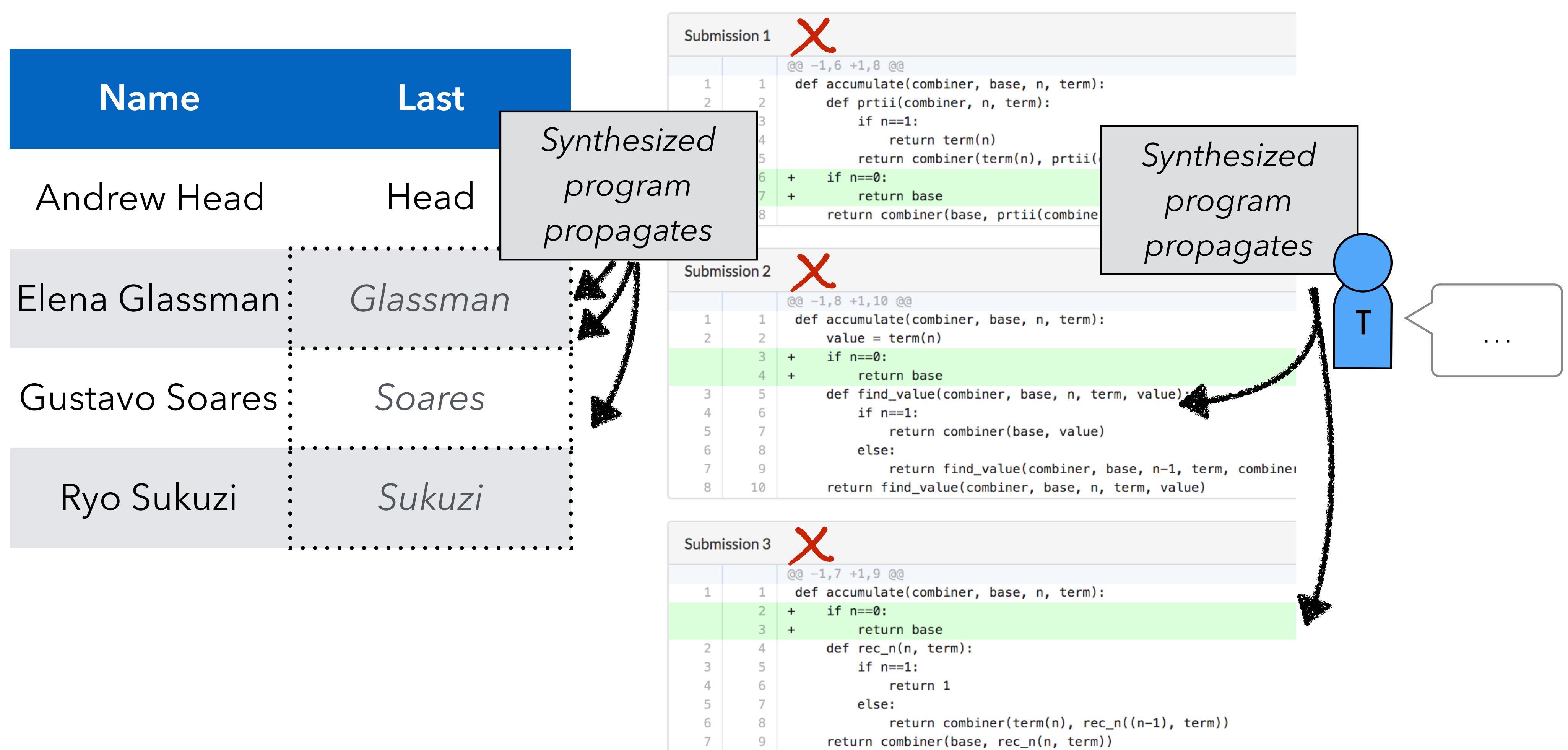
Human demonstrates

Submission 1 

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

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

Propagating Transformations



What Bug-Fixing Code Transformations Can Refazer Learn?

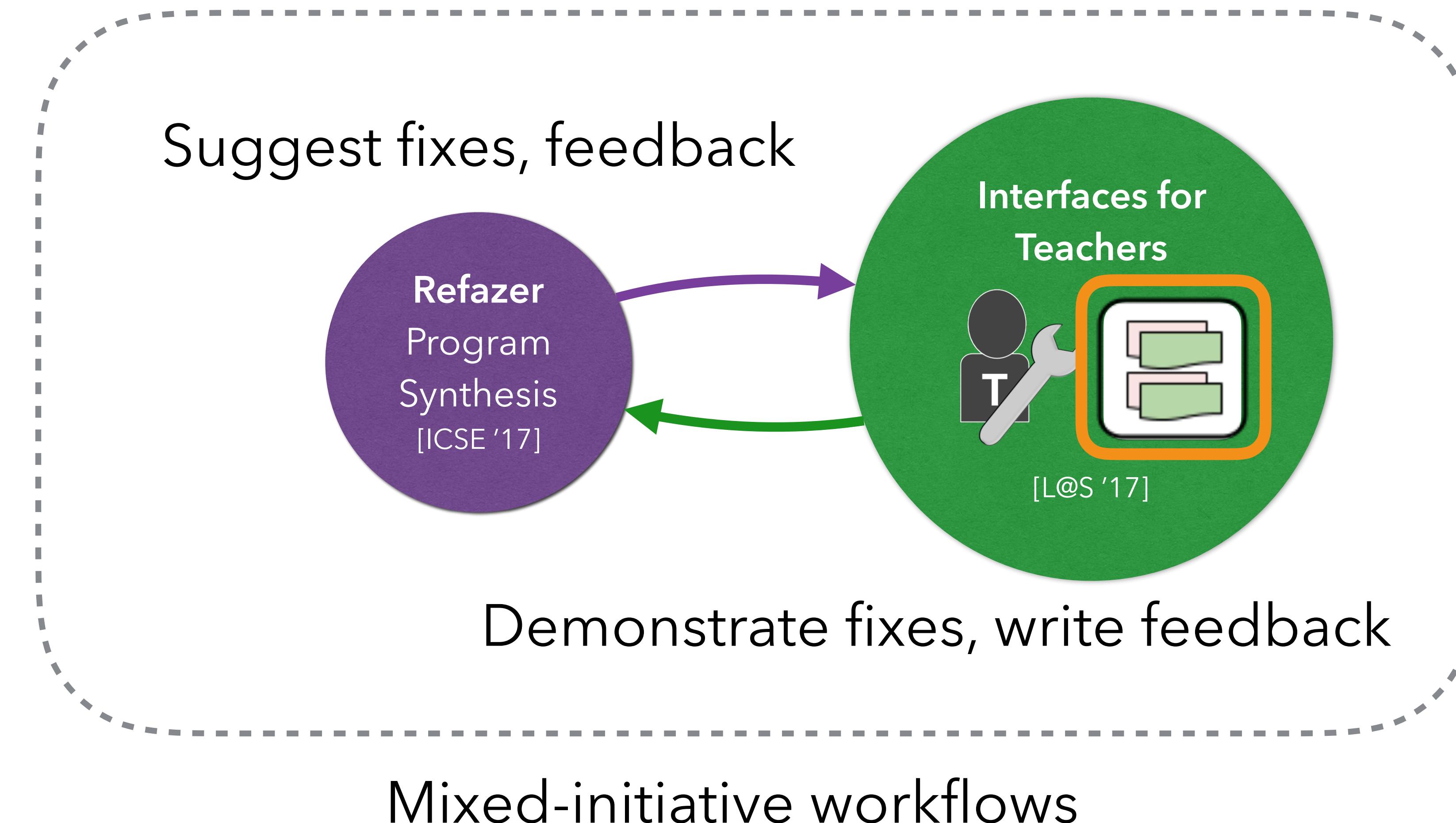
```
1  1  def accumulate(combiner, base, n, term):  
2  2      def prtii(combiner, n, term):  
3  3          if n==1:  
4  4              return term(n)  
5  5          return combiner(term(n), prtii(combiner, n-1, term))  
6  +  if n==0:  
7  +      return base  
8  8  return combiner(base, prtii(combiner,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

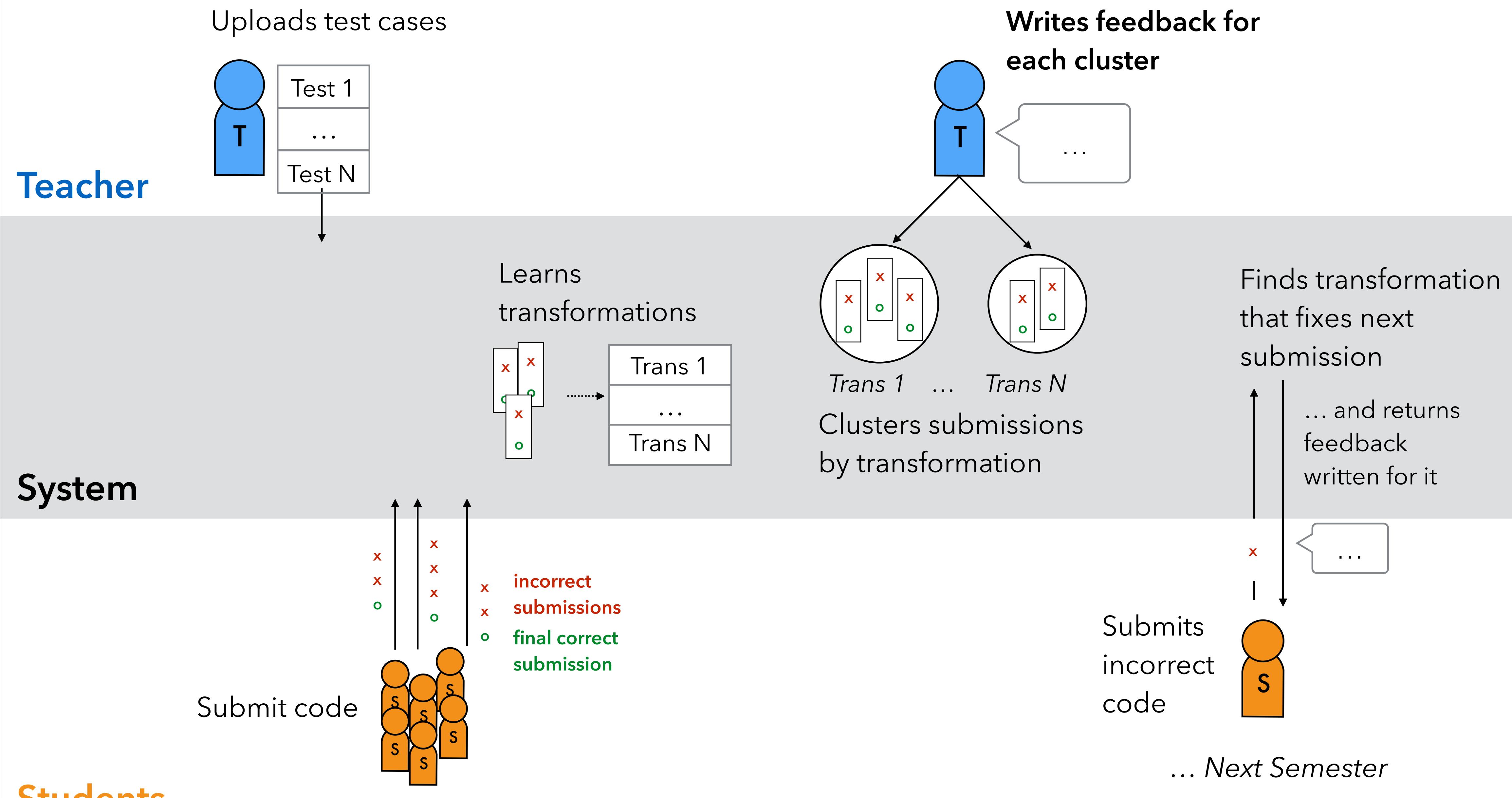
Missing base cases...

```
1  1  def repeated(f, n):  
2  2      if n==0:  
3  -  return f  
3  +  return identity  
4  4  else:  
5  5      return compose1(f, repeated(f, n-1))  
6  6  def compose1(f, g):  
7  7      def h(x):  
8  8          return f(g(x))  
9  9  return h
```

Function substitutions... and so on...

System Design





Assignment description

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

```
n -- a positive integer
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
```

Cluster

Cluster 1

41

Examples of applied fix

- return term(n)*term(n-1)

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

Submissions

Select all submissions

Submission 1

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

Test feedback

Input	Expected	Actual
product(5, identity)	120	20

Submission 2

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

Test feedback

Input	Expected	Actual
product(5, identity)	120	20

Submission 3

```
@@ -1,5 +1,5 @@
1 1 def product(n, term):
2 2     if n<=1:
```

Hints

Set Hint

Reuse previous hints

Assignment description

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

n -- a positive integer
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
```

Cluster

Cluster 1

41

Examples of applied fix

- return term(n)*term(n-1)

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

Submissions

Select all submissions

Submission 1

1	1	@@ -1,5 +1,5 @@
2	2	def product(n, term):
3	3	if n<=1:
4	4	return 1
5	5	else:
		- return term(n)*term(n-1)
		+ return term(n)*product(n-1, term)

Test feedback

Input	Expected	Actual
product(5, identity)	120	20

Submission 2

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

Test feedback

Input	Expected	Actual
product(5, identity)	120	20

Submission 3

1	1	@@ -1,5 +1,5 @@
2	2	def product(n, term):
		if n==1:

Hints

Set Hint

Reuse previous hints

Assignment description

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

```
n -- a positive integer
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
```

Cluster

Cluster 1

41

Examples of applied fix

- return term(n)*term(n-1)

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

Submissions

Select all submissions

Submission 1

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

Test feedback

Input	Expected	Actual
product(5, identity)	120	20

Submission 2

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

Test feedback

Input	Expected	Actual
product(5, identity)	120	20

Submission 3

```
@@ -1,5 +1,5 @@
1   1 def product(n, term):
2   2     if n<=1:
```

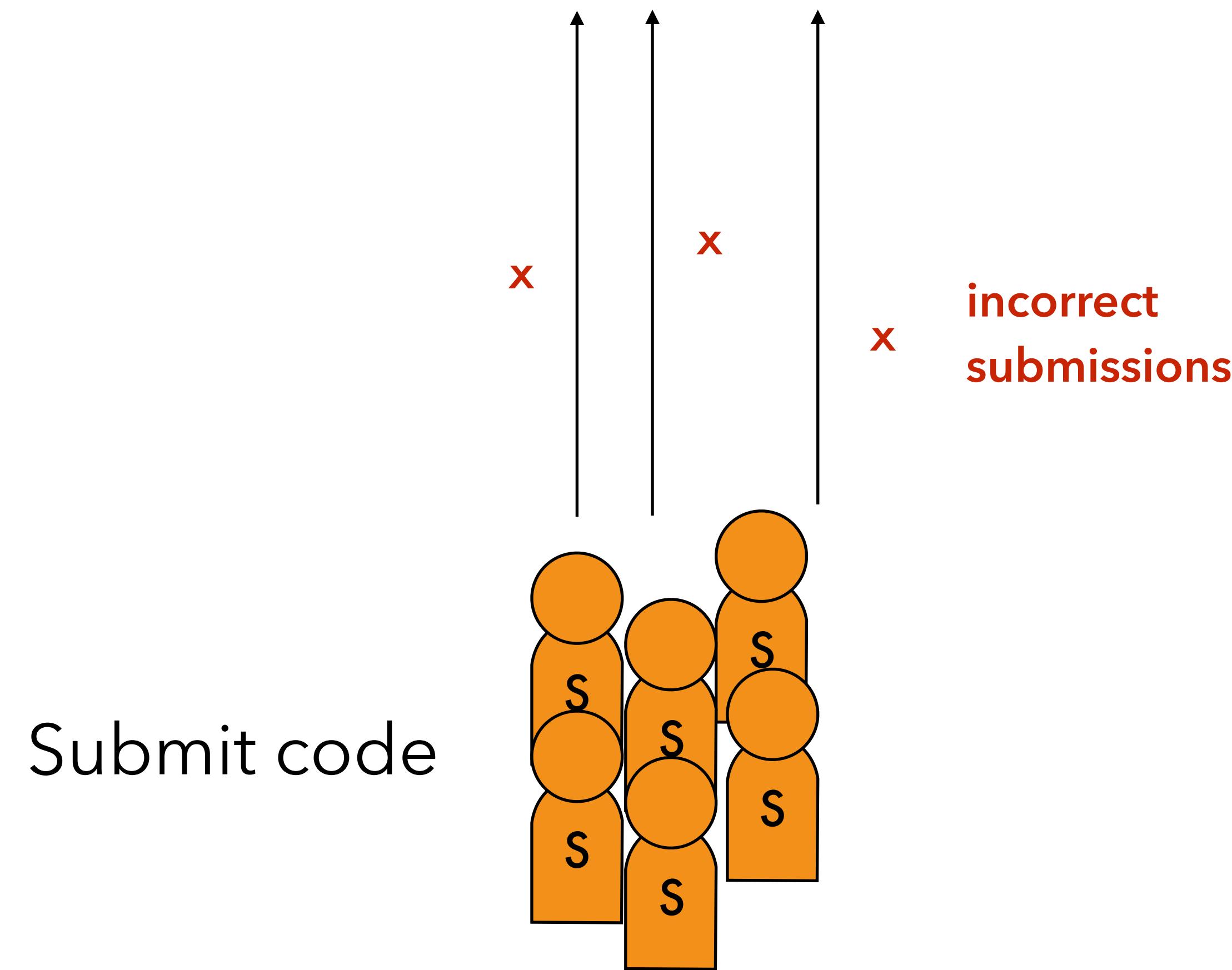
Hints

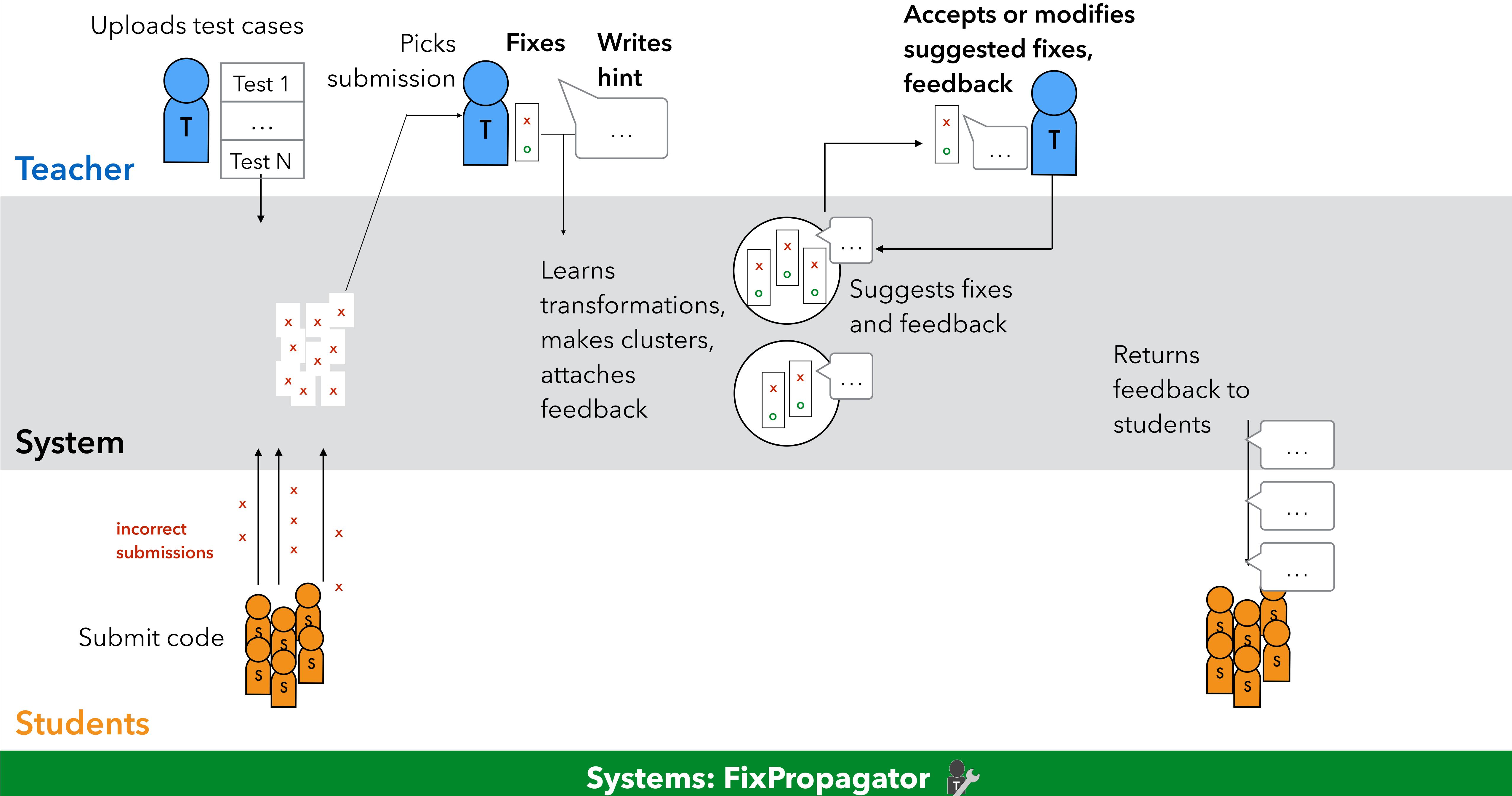
Looks like you're writing a recursive call. What might you be missing to enable recursion?

[Set Hint](#)

[Reuse previous hints](#)

But Not All Classes Have Submission Histories for Hundreds of Students





Submissions

= feedback given
 = passed all test cases
 = fix suggested

- [Submission 281](#)
- Submission 282
- Submission 283
- Submission 284
- Submission 285
- Submission 286
- Submission 287
- Submission 288
- Submission 289
- Submission 290
- Submission 291
- Submission 292
- Submission 293

Order by:

- Submission IDs
- Test case results
- Suggested fixes

Suggested fixes

Student Submission

You can edit this code. Show original Edit Show diff

```
1 def product(n, term):  
2     |  
3         return term(n) * product(n - 1, term)  
4
```

Run tests again

Test results: Some tests failed

Test	Input	Result	Expected	Output
1	(3, lambda x: x),	→ RecursionError	6	
2	(5, lambda x: x),	→ RecursionError	120	
3	(3, lambda x: x * x),	→ RecursionError	36	
4	(5, lambda x: x * x),	→ RecursionError	14400	

Print output (test case 1)

RecursionError: ('maximum recursion depth exceeded',)

[This test case produced no console output.]

Feedback

Notes

Add

Submit feedback

Submissions

= feedback given
 = passed all test cases
 = fix suggested

Submission 281

Submission 282

Submission 283

Submission 284

Submission 285

Submission 286

Submission 287

Submission 288

Submission 289

Submission 290

Submission 291

Submission 292

Submission 293

Order by:

- Submission IDs
- Test case results
- Suggested fixes

Suggested fixes

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     return term(n) * product(n - 1, term)  
5
```

Run tests again

Test results: All tests succeeded

Test	Input	Result	Expected	Output
1	(3, lambda x: x),	→ 6	6	
2	(5, lambda x: x),	→ 120	120	
3	(3, lambda x: x * x),	→ 36	36	
4	(5, lambda x: x * x),	→ 14400	14400	

Print output (test case 1)

[This test case produced no console output.]

Feedback

Notes

Add

Submit feedback

New Student Submission with Same Bug

Suggested Fix

Submissions

= feedback given
 = passed all test cases
 = fix suggested

- [Submission 146](#)
- Submission 147
- Submission 148
- Submission 149
- Submission 150
- Submission 151
- Submission 152
- Submission 153
- Submission 154
- Submission 155
- Submission 156
- Submission 157
- Submission 158

Order by:

- Submission IDs
- Test case results
- Suggested fixes

Suggested fixes

- [Submission 21](#)
- Submission 24
- Submission 25

Student Submission

You can edit this code. Show original Edit Show diff

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

Run tests again

Test results: Some tests failed

Test	Input	Result	Expected	Output
1	(3, lambda x: x),	→ TypeError	6	
2	(5, lambda x: x),	→ TypeError	120	
3	(3, lambda x: x * x),	→ TypeError	36	
4	(5, lambda x: x * x),	→ TypeError	14400	

Print output (test case 1)

```
TypeError: ("unsupported operand type(s) for *: 'int' and 'NoneType'",)
```

```
[This test case produced no console output.]
```

Back

Next

Feedback

Student error detected.

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

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

← 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

Add

Submit feedback

Submissions

= feedback given
 = passed all test cases
 = fix suggested

Submission 146	
Submission 147	
Submission 148	
Submission 149	
Submission 150	
Submission 151	
Submission 152	
Submission 153	
Submission 154	
Submission 155	
Submission 156	
Submission 157	
Submission 158	

Order by:

- Submission IDs
- Test case results
- Suggested fixes

Suggested fixes

Submission 24	
Submission 25	

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)  
6
```

Run tests again

Test results: All tests succeeded

Test	Input	Result	Expected	Output
1	(3, lambda x: x),	→ 6	6	
2	(5, lambda x: x),	→ 120	120	
3	(3, lambda x: x * x),	→ 36	36	
4	(5, lambda x: x * x),	→ 14400	14400	

Print output (test case 1)

[This test case produced no console output.]

Back

Next

Feedback

Student error detected.

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

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

← 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

Add

Submit feedback

Submissions

= feedback given
 = passed all test cases
 = fix suggested

- Submission 146
- Submission 147
- Submission 148
- Submission 149
- Submission 150
- Submission 151
- Submission 152
- Submission 153
- Submission 154
- Submission 155
- Submission 156
- Submission 157
- Submission 158

Order by:

- Submission IDs
- Test case results
- Suggested fixes

Suggested fixes

- Submission 24
- Submission 25

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)
```

Test results: All tests

Test	Input
1	(3, lambda
2	(5, lambda
3	(3, lambda
4	(5, lambda x: x * x),

Both Fixes and
Feedback Can Be
Further Modified

Print output (test case 1)

[This test case produced no console output.]

Back

Next

Feedback

Student error detected.

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

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

[← 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

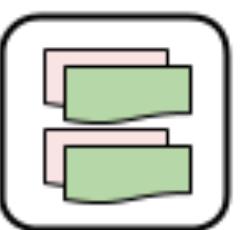
 Add

What should happen when $n == 0$? X

[Submit feedback](#)

A Study of the Systems

Participants: Current and former teaching staff from CS1



MistakeBrowser ($N = 9$)



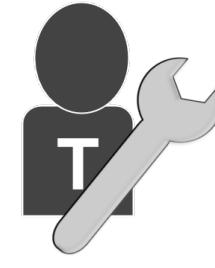
FixPropagator ($N = 8$)

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?

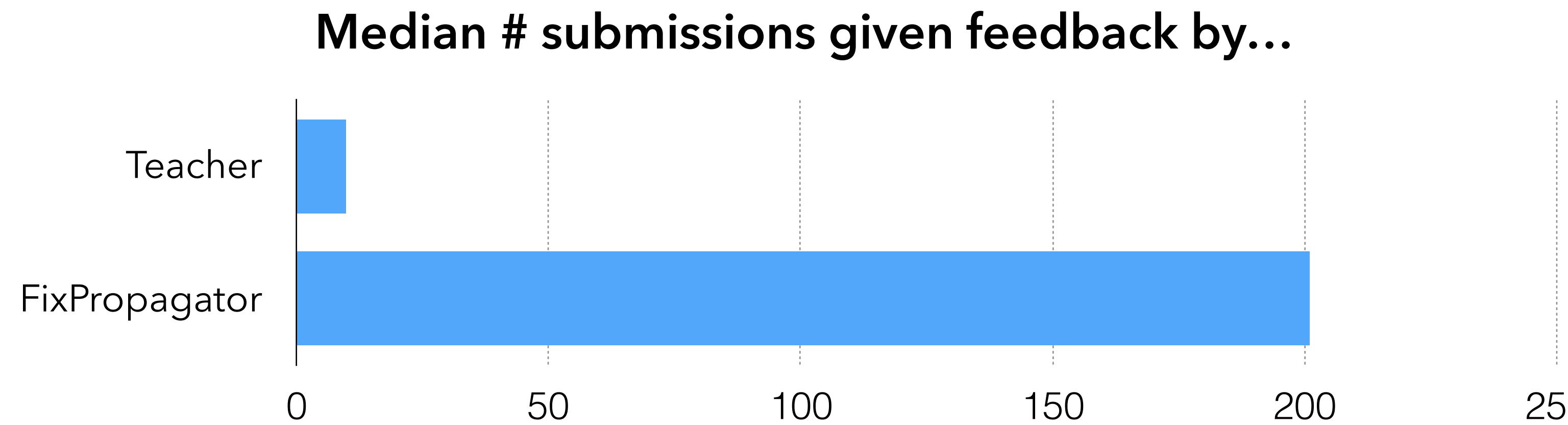


FixPropagator propagates fixes from dozens of corrections to hundreds of submissions.

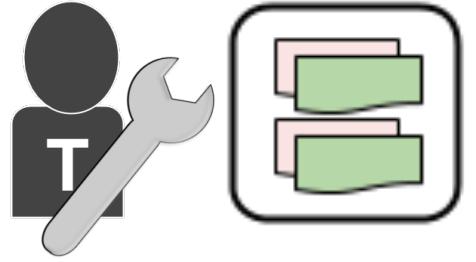
1. Can a few manual corrections fix many submissions?



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



- Fixes were propagated within minutes (*median = 2m20s, $\sigma = 7m34s$* for each correction).



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

- 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%).

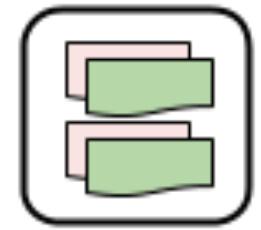
**Generalizable
Comment**

"Check if you have the product of the correct number of terms."

**Non-Generalizable
Comment**

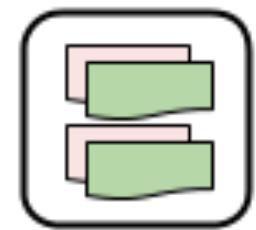
"Your starting value of z should be a function, not an int."

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

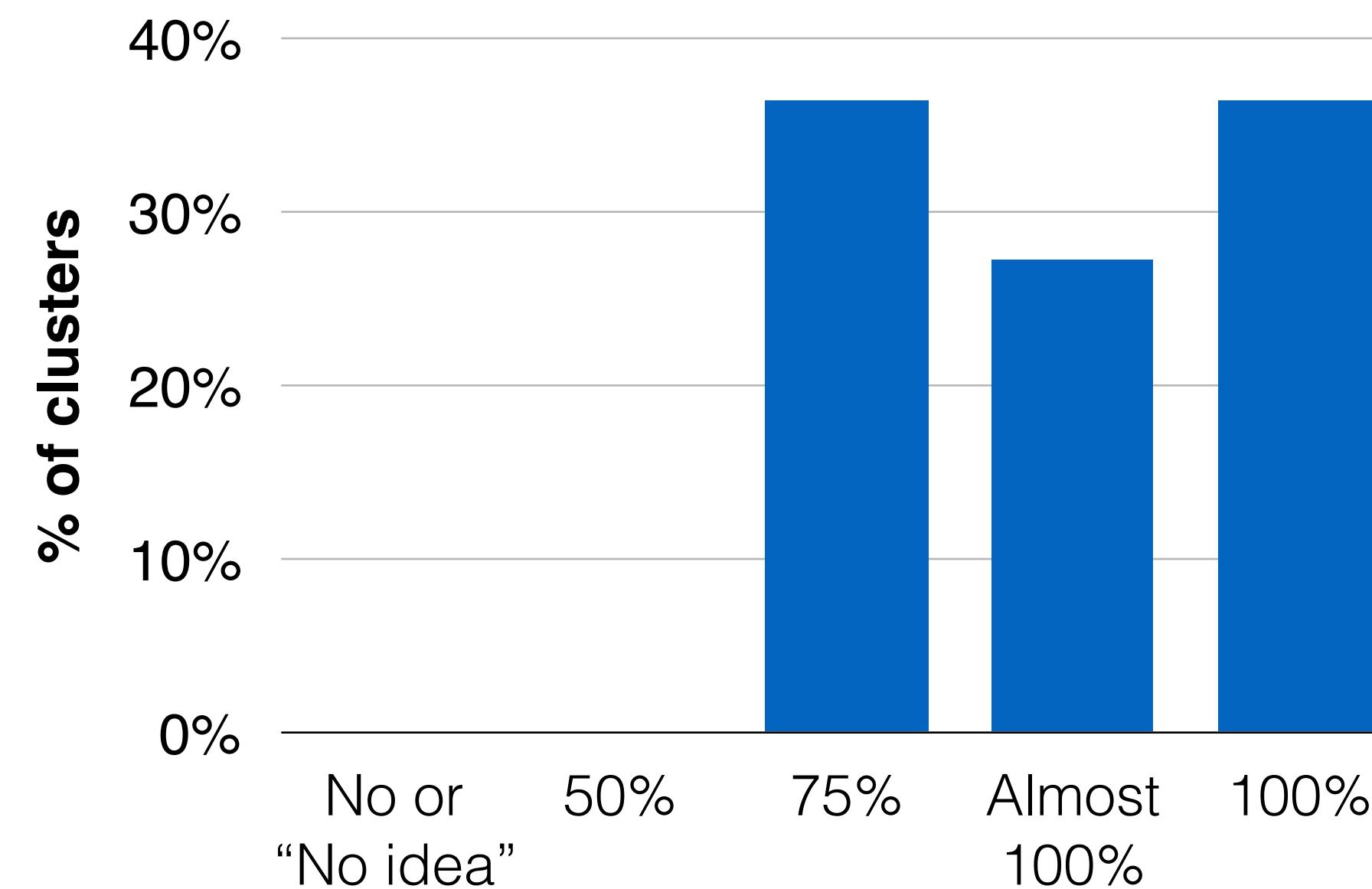


MistakeBrowser created conceptually consistent clusters of student bugs.

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.



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).

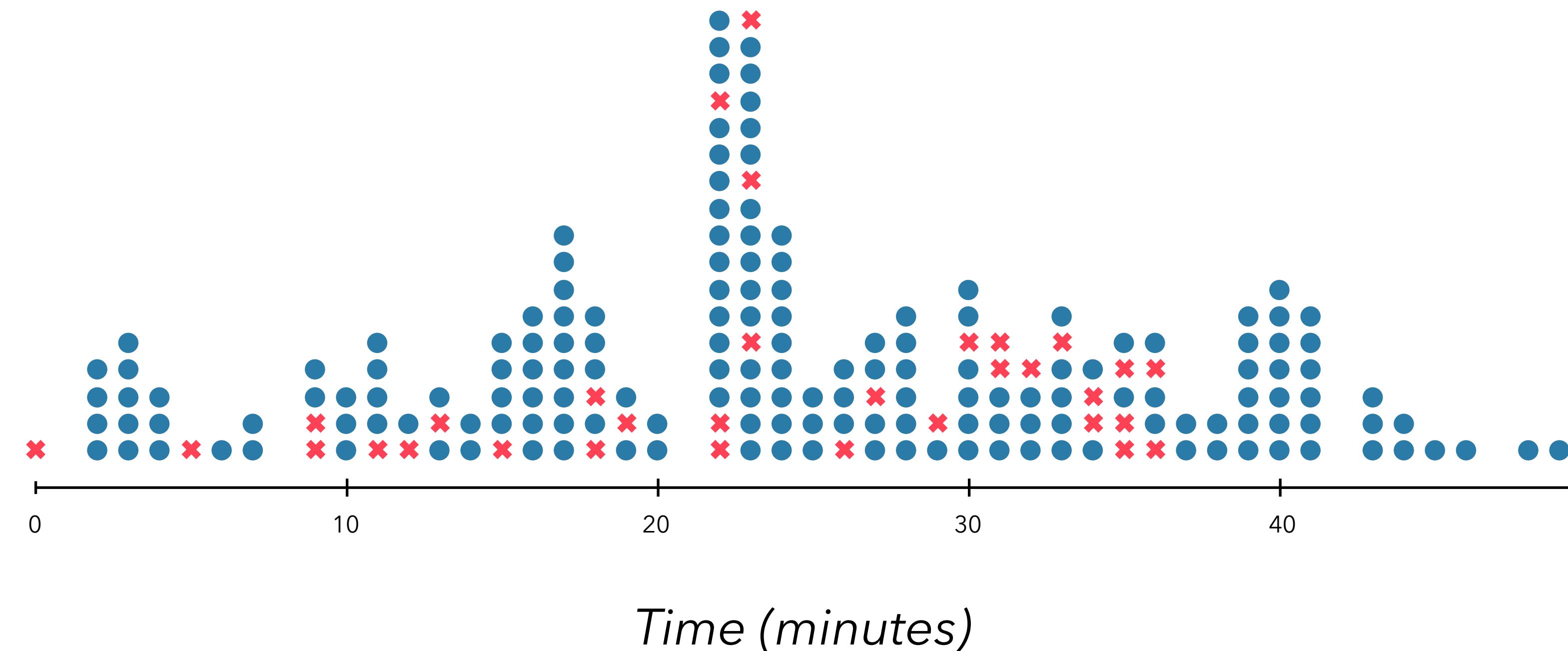
Seeing all of the similar instances of the same (or nearly the same) misconception was very useful, because it suggested ways to address common issues shared by many students.

- Participant 3, about MistakeBrowser

1. Can a few manual corrections fix many submissions?



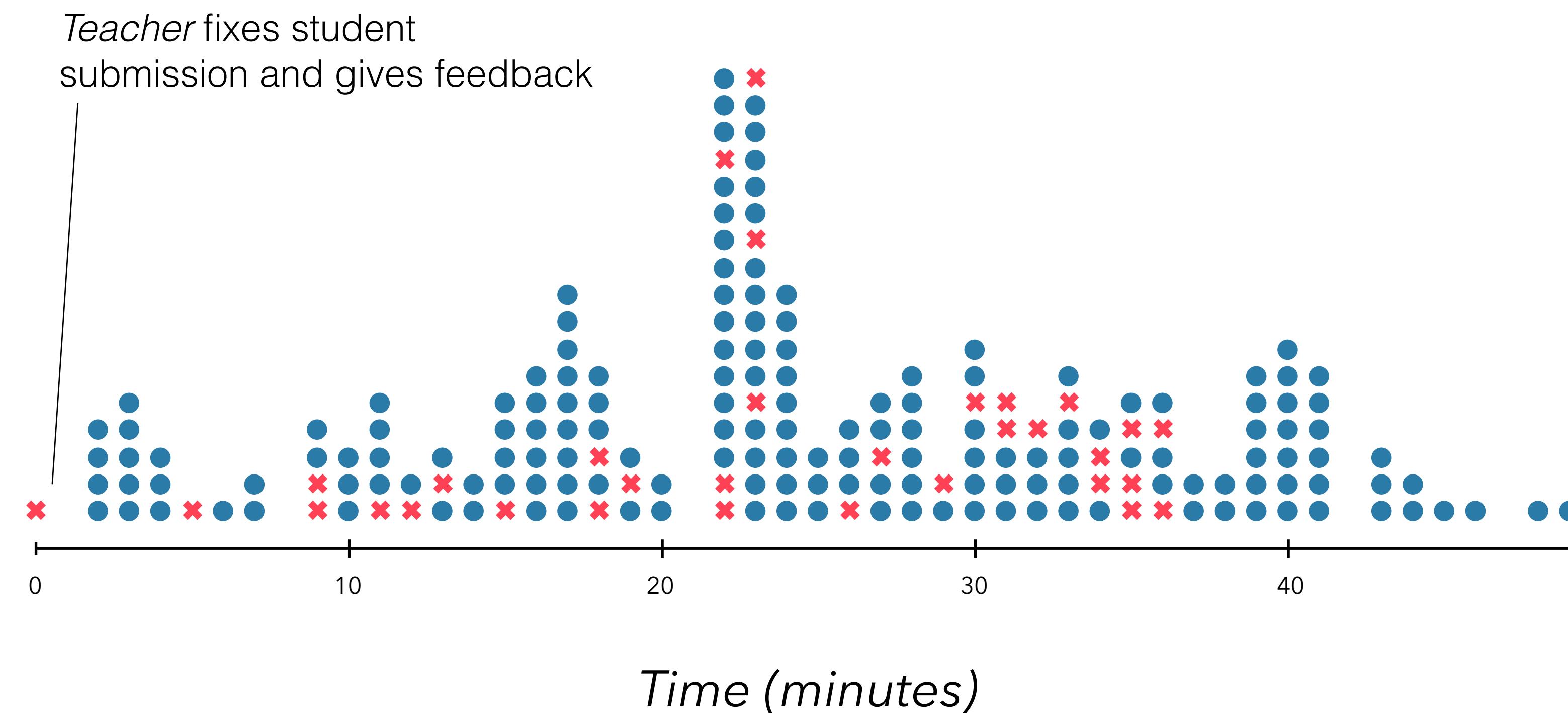
FixPropagator propagates fixes from dozens of corrections to hundreds of submissions.



1. Can a few manual corrections fix many submissions?



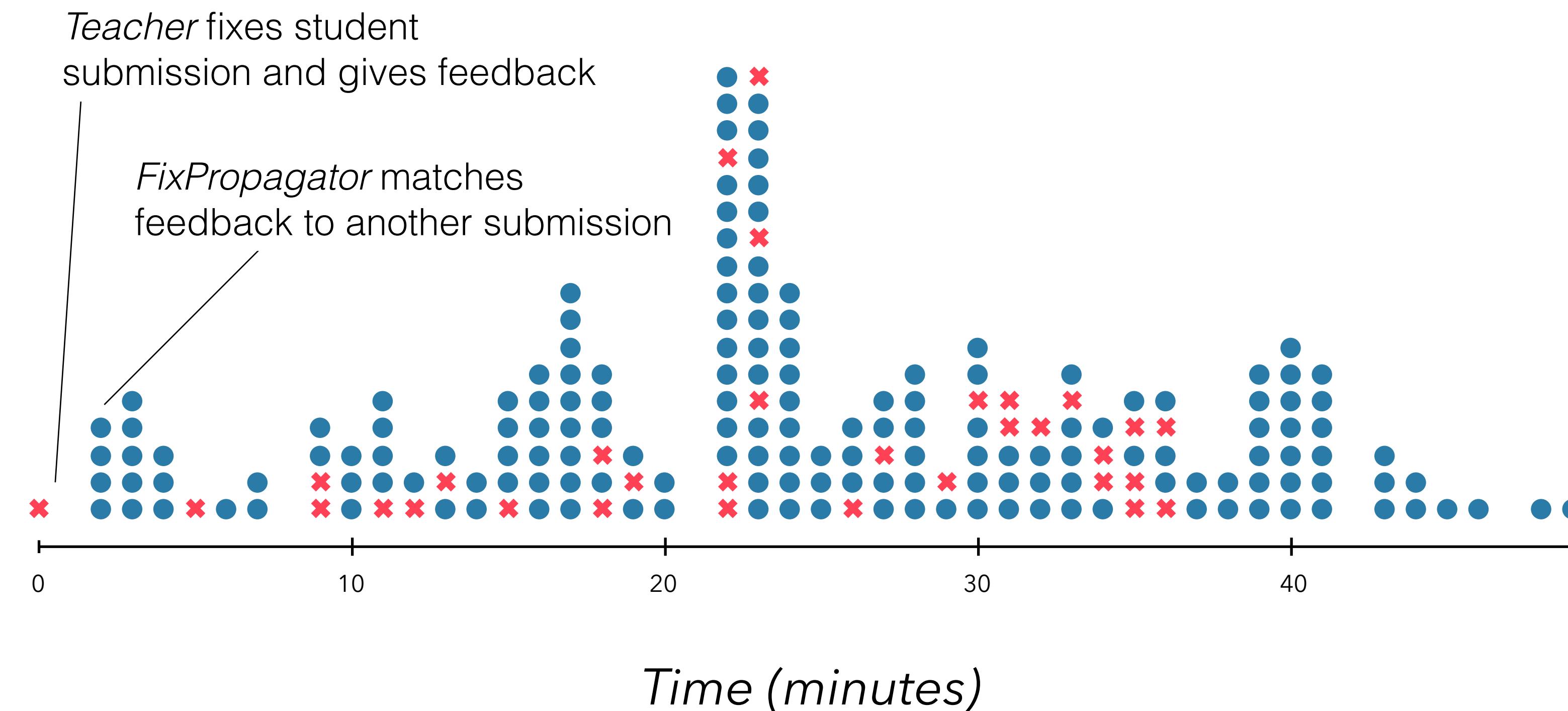
FixPropagator propagates fixes from dozens of corrections to hundreds of submissions.



1. Can a few manual corrections fix many submissions?



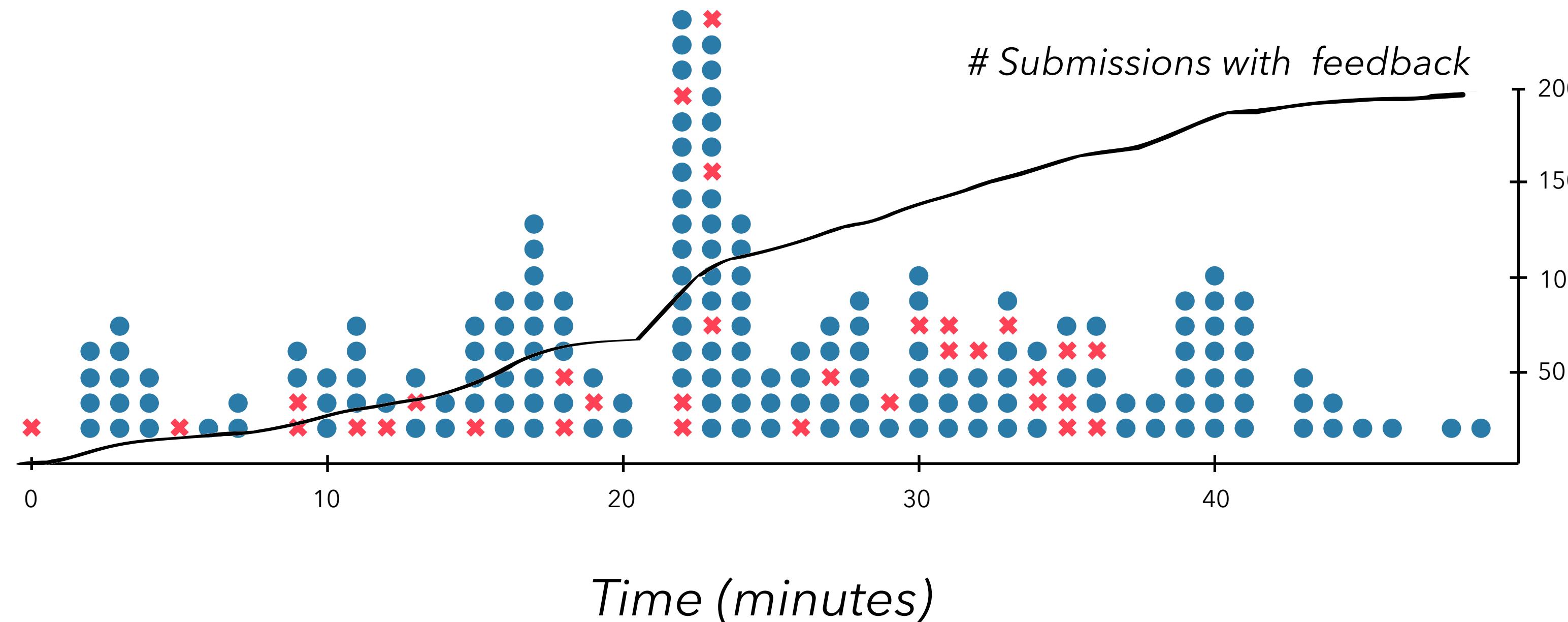
FixPropagator propagates fixes from dozens of corrections to hundreds of submissions.



1. Can a few manual corrections fix many submissions?



FixPropagator propagates fixes from dozens of corrections to hundreds of 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.

- 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.

This wrong answer can be "fixed" with the edits for [submission 24](#).

This is the fix:

```
@@ -1,6 +1,5 @@
1 -
2 1 def product(n, term):
3 -    total, k = 0, 1
2+ total, k = 1, 1
4 3 while k <= n:
5 4     total, k = total * term(k)
6 5 return total
```

[← Apply this fix to the student's code](#)

- 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.

This wrong answer can be "fixed" with the edits for [submission 24](#).

This is the fix:

```
@@ -1,6 +1,5 @@
1 -
2 1 def product(n, term):
3 -    total, k = 0, 1
2+ total, k = 1, 1
4 3 while k <= n:
5 4     total, k = total * term(k)
6 5 return total
```

← Apply this fix to the student's code



"Propagating" a fix

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.

This wrong answer can be "fixed" with the edits for [submission 24](#).

This is the fix:

```
@@ -1,6 +1,5 @@
1 -
2 1 def product(n, term):
3 -    total, k = 0, 1
2+ total, k = 1, 1
4 3    while k <= n:
5 4        total, k = total * term(k), k + 1
6 5    return total
```

[← Apply this fix to the student's code](#)

You can edit this code. Show original Edit Show diff

```
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
6
```

[Run tests again](#)

“Propagating” a fix

- 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.

This wrong answer can be "fixed" with the edits for [submission 24](#).

This is the fix:

```
@@ -1,6 +1,5 @@
1 -
2 1 def product(n, term):
3 -    total, k = 0, 1
2+ total, k = 1, 1
4 3 while k <= n:
5 4     total, k = total * term(k)
6 5 return total
```

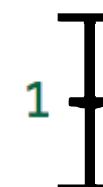
[← Apply this fix to the student's code](#)



"Propagating" a fix

You can edit this code. Show original Edit Show diff

```
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
6
```

 "Modifying" a fix

[Run tests again](#)

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.

This wrong answer can be "fixed" with the edits for [submission 24](#).

This is the fix:

```
@@ -1,6 +1,5 @@
1 -
2 1 def product(n, term):
3 -    total, k = 0, 1
2+ total, k = 1, 1
4 3 while k <= n:
5 4     total, k = total * term(k)
6 5 return total
```

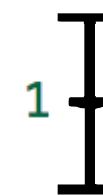
[← Apply this fix to the student's code](#)



"Propagating" a fix

You can edit this code. Show original Edit Show diff

```
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
6
```

 "Modifying" a fix

[Run tests again](#)

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?

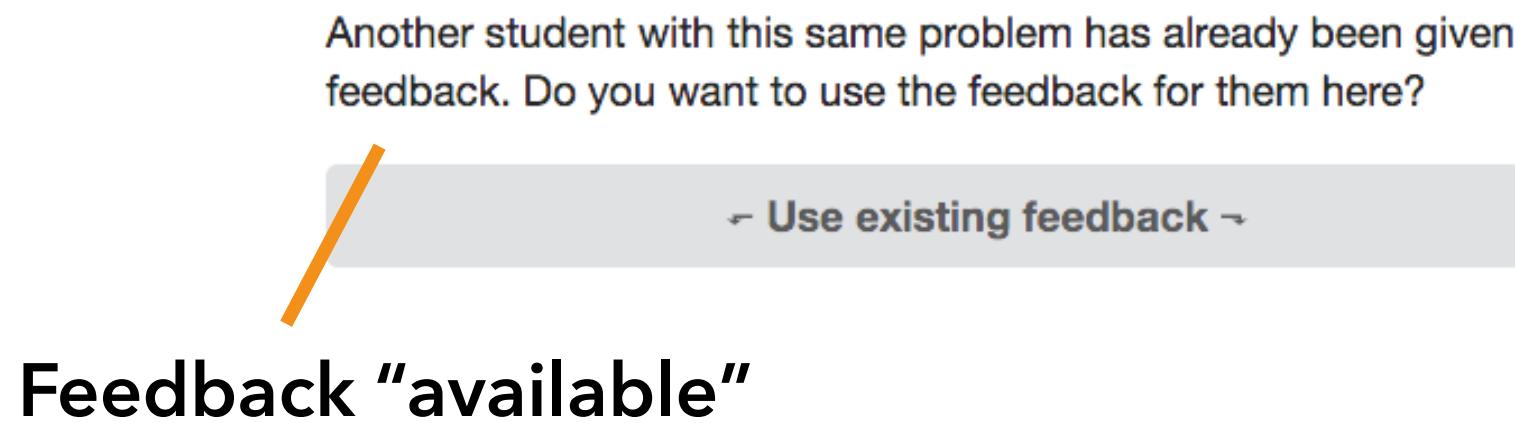
Feedback propagated with FixPropagator was correct a majority of the time, but not always.

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

→ Use existing feedback →

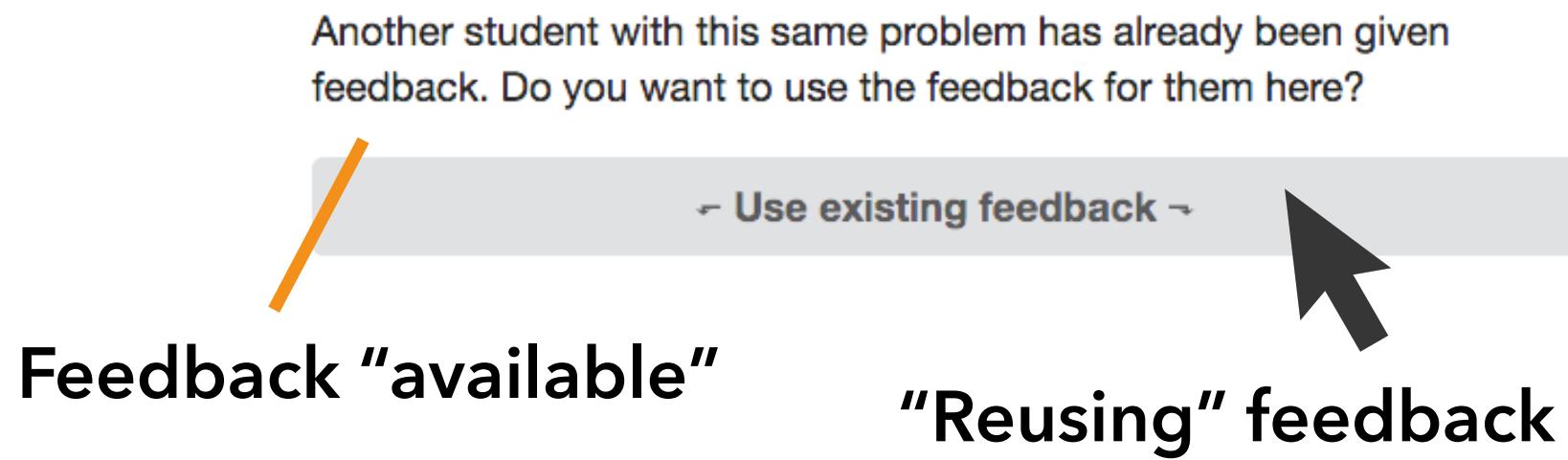
2. How often is a teacher's feedback relevant when it is matched to other students' submission?

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.

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

Use existing feedback

Feedback "available" "Reusing" feedback

Notes

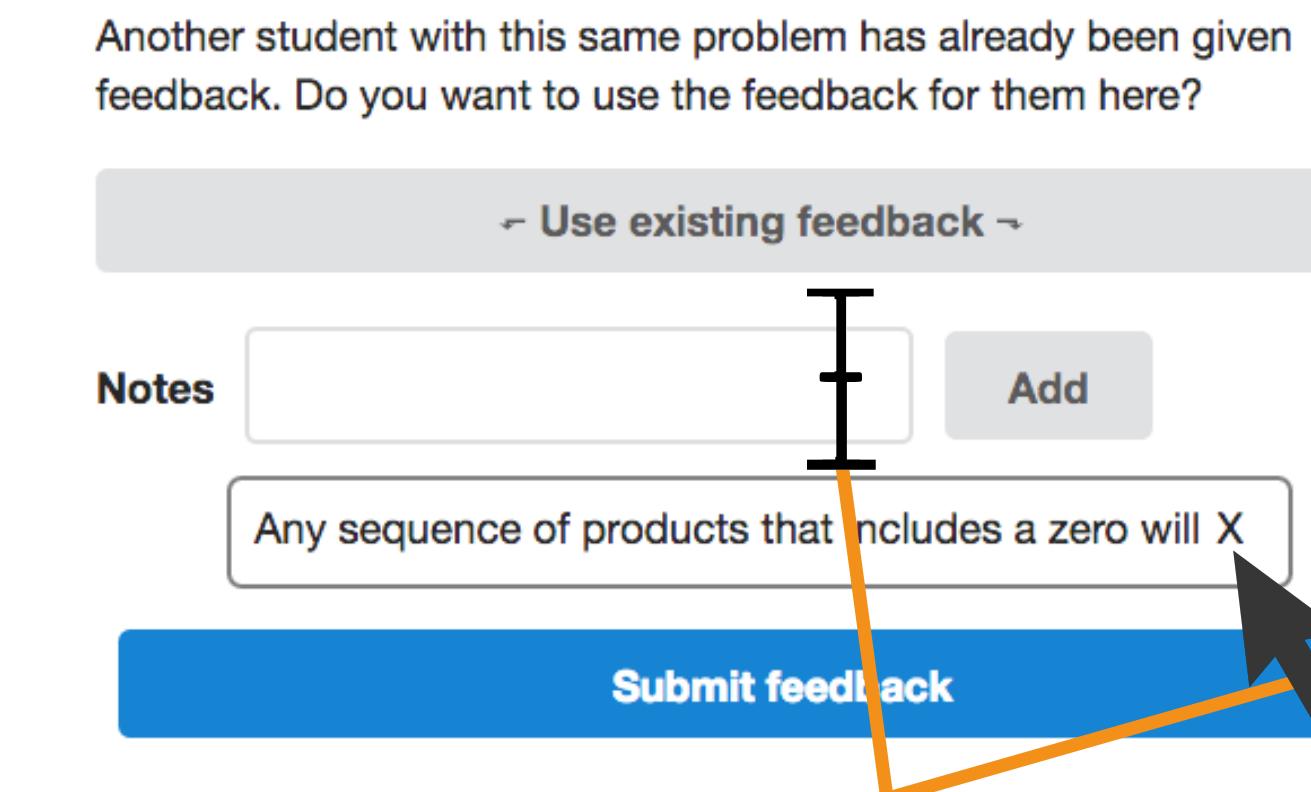
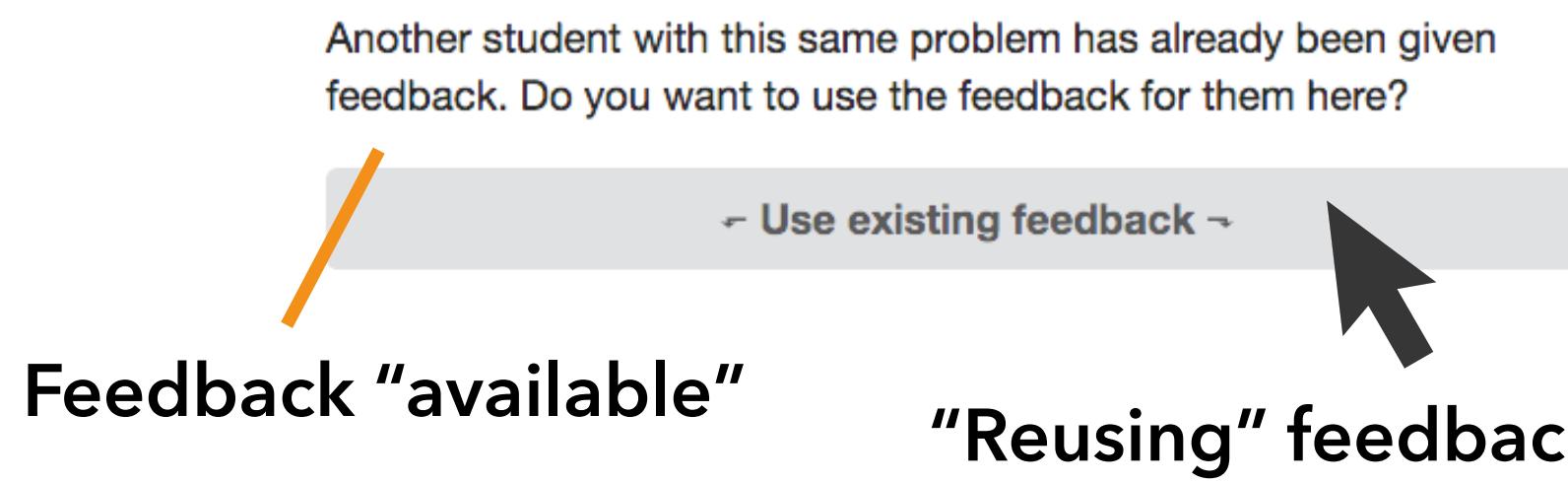
Any sequence of products that includes a zero will X

Add

Submit feedback

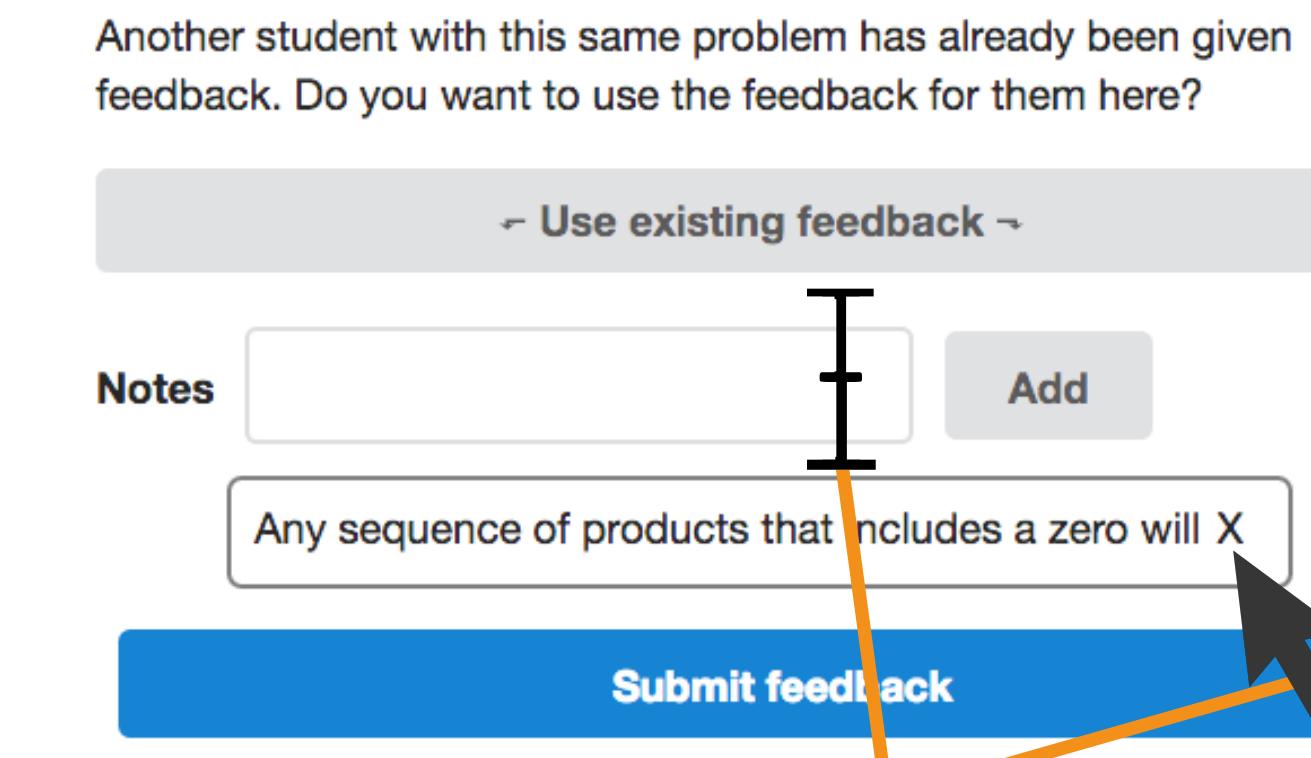
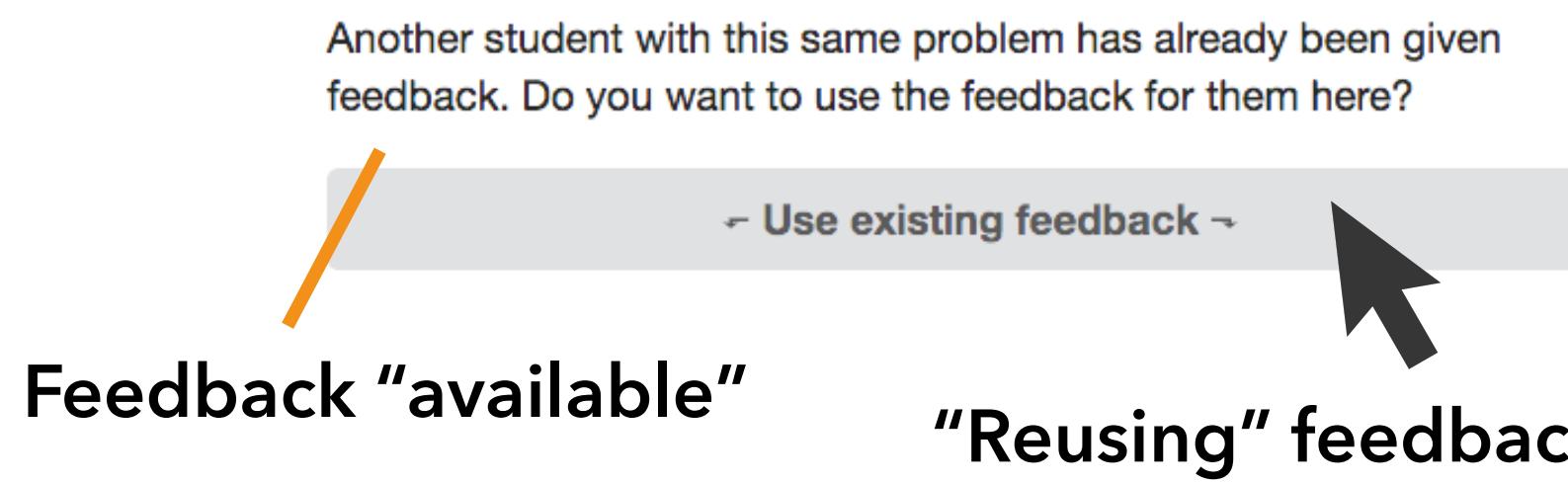
- How often is a teacher's feedback relevant when it is matched to other students' submission?

Feedback propagated with FixPropagator was correct a majority of the time, but not always.



- 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.

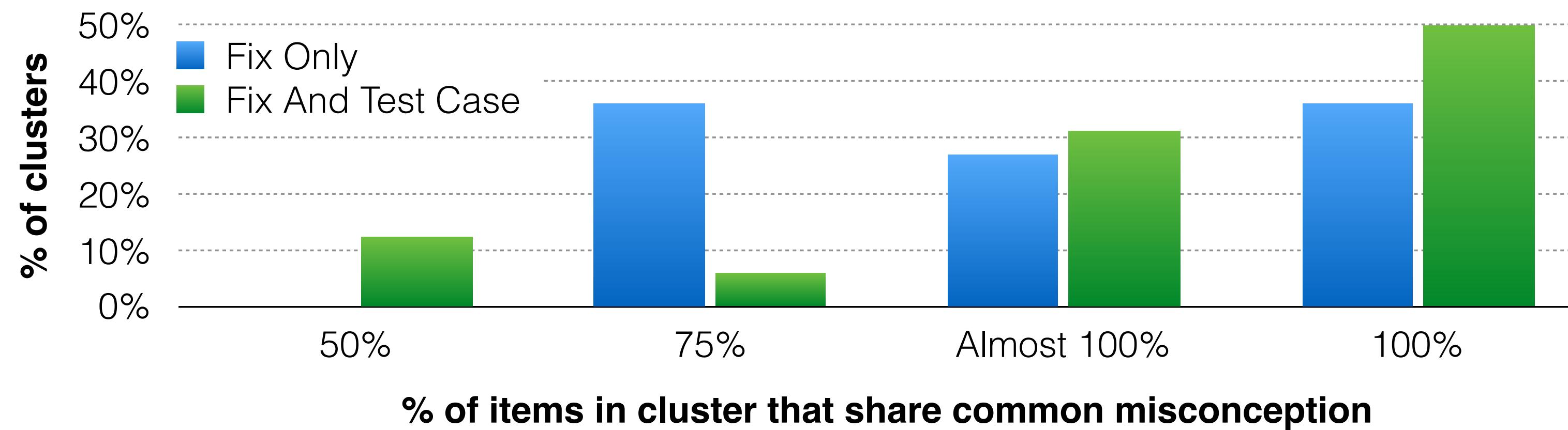


2. How often is a teacher's feedback relevant when it is matched to other students' submission?

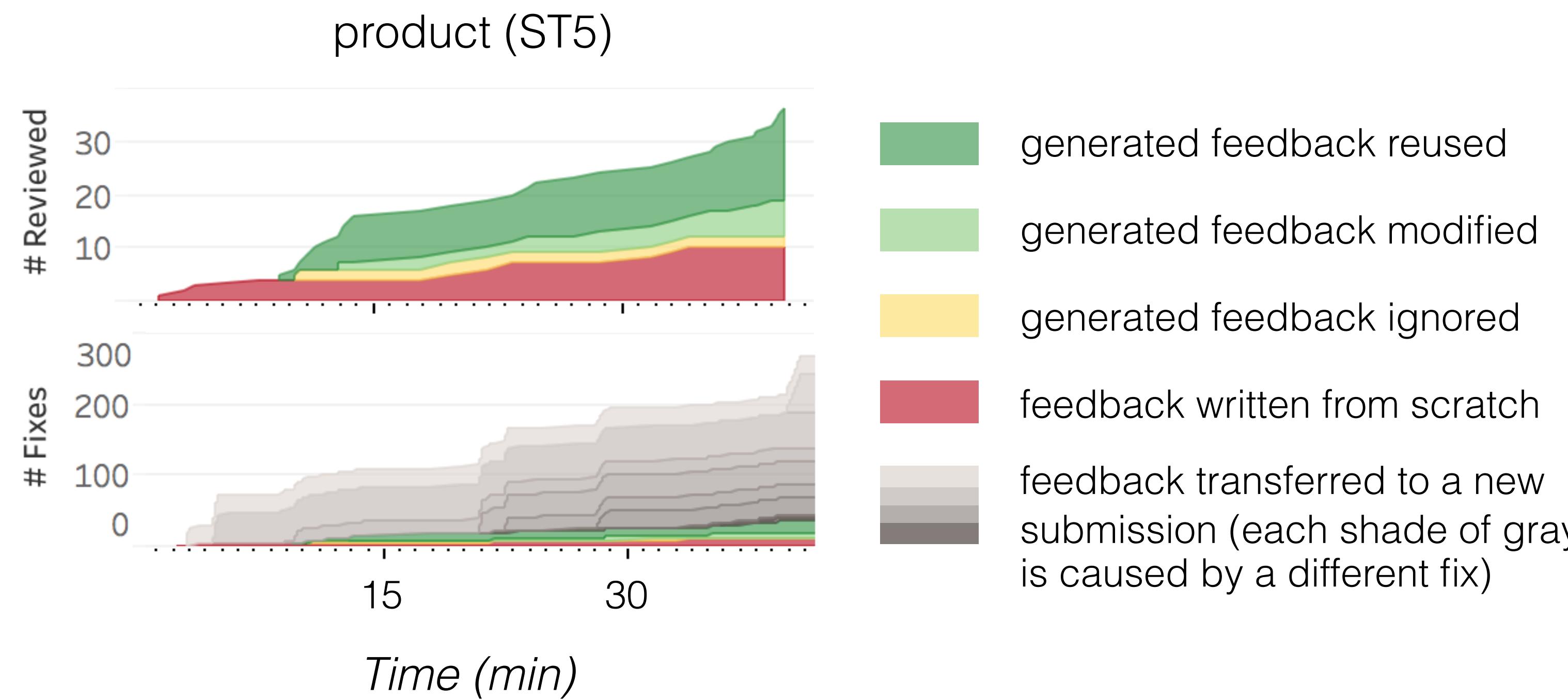
Feedback propagated with FixPropagator was correct a majority of the time, but not always.

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



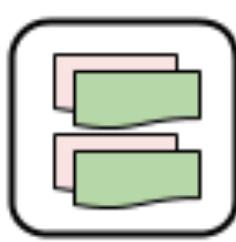
Limitations

- 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

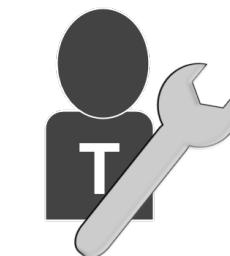
Conclusion

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

And two systems implementing this approach:



MistakeBrowser

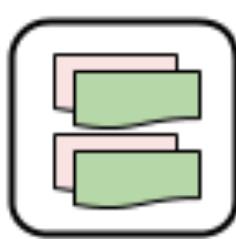


FixPropagator

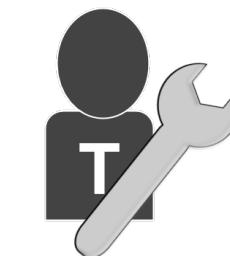
Conclusion

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?