• Test-cases based feedback
  • Hard to relate failing inputs to errors

• Manual feedback by TAs
  • Time consuming and error prone
"Not only did it take 1-2 weeks to grade problem, but the comments were entirely unhelpful in actually helping us fix our errors. ..... Apparently they don't read the code – they just ran their tests and docked points mercilessly. What if I just had a simple typo, but my algorithm was fine? ...."
Scalability Challenges (>100k students)

Bigger Challenge in MOOCs
Today's Grading Workflow

Teacher's Solution
Grading Rubric

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

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

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

Teacher's Solution  Error Model

replace derive by [0]
Technical Challenges

Large space of possible corrections
Minimal corrections
Dynamically-typed language

Constraint-based Synthesis to the rescue
Running Example
**computeDeriv**

Compute the derivative of a polynomial

\[ \text{poly} = [10, 8, 2] \quad \# f(x) = 10 + 8x + 2x^2 \]

\[ \Rightarrow [8, 4] \quad \# f'(x) = 8 + 4x \]
def computeDeriv(poly):
    result = []
    if len(poly) == 1:
        return [0]
    for i in range(1, len(poly)):
        result += [i * poly[i]]
    return result
Demo:

Simplified Error Model

- return a \rightarrow return \{[0], ?a\}
- range(a_1, a_2) \rightarrow range(a_1+1, a_2)
- a_0 == a_1 \rightarrow False
Autograder Algorithm
Algorithm

Rewriter

Translator

Solver

Feedback

.py

.py

.sk

.out
Algorithm: Rewriter
Rewriting using Error Model

\[ \text{range}(0, \text{len}(\text{poly})) \]

\[ \text{range}(\{0, 1\}, \text{len}(\text{poly})) \]

default choice

\[ a \rightarrow a + 1 \]
Rewriting using Error Model

\[ \text{range}(0, \text{len(poly)}) \]

\[ \text{range}([0, 1], \text{len(poly)}) \]

\[ a \rightarrow a+1 \]
Rewriting using Error Model

\[ \text{range}(0, \text{len}(\text{poly})) \]

\[ \xrightarrow{\text{a} \rightarrow \text{a+1}} \]

\[ \text{range}(\{0,1\}, \text{len}(\{\text{poly}, \text{poly+1}\})) \]
Rewriting using Error Model

\[ \text{range}(0, \text{len}(\text{poly})) \]

\[ \text{range}([0, 1], \{\text{len}([\text{poly}, \text{poly+1}]), \text{len}([\text{poly}, \text{poly+1}])+1\}) \]

\[ a \rightarrow a+1 \]
def computeDeriv(poly):
    deriv = []
    zero = 0
    if ({len(poly) == 1, False}):
        return {deriv, [0]}
    for e in range({0, 1}, len(poly)):
        if (poly[e] == 0):
            zero += 1
        else:
            deriv.append(poly[e] * e)
    return {deriv, [0]}

Problem: Find a program that minimizes cost metric and is functionally equivalent with teacher’s solution
Algorithm: Translator

Rewriter → Translator → Solver → Feedback

.\text{py} \quad \text{.sk}
A Synthesis Primer

The Synthesis problem as a doubly quantified constraint

$$\exists P \forall in \ (in, P \models Spec)$$

- What does it mean to quantify over programs?
Quantifying over programs

Synthesis as curve fitting

It’s hard to do curve fitting with arbitrary curves

- Instead, people use parameterized families of curves
- Quantify over parameters instead of over functions

\[ \exists c \forall i \in \text{in} \quad (\text{in}, P[c] \models \text{Spec}) \]

Key idea:
Let user define parameterized functions with partial programs
Sketch [Solar-Lezama et al. ASPLOS06]

void main(int x){
    int k = ??;
    assert x + x == k * x;
}

void main(int x){
    int k = 2;
    assert x + x == k * x;
}

Statically typed C-like language with holes
Py Translation to Sketch

(1) Handling python’s dynamic types

(2) Translation of expression choices
Algorithm: Solver

Rewriter → Translator → Solver → Feedback

Solver: .sk
Feedback: .out
CEGIS Synthesis algorithm

\[ \exists c \text{ s.t. } \text{Correct}(P_c, in_i) \]

\[ \exists \text{in s.t. } \neg \text{Correct}(P_c, in_i) \]
CEGIS

\[ Q(c, \text{in}) \]

Synthesize

\[
\begin{align*}
Q(c, in_0) & \quad Q(c, in_1) \\
\neg Q(c, in_2) & \quad Q(c, in_3)
\end{align*}
\]

Check

\[ \neg Q(c, in_2) \]
Algorithm: Feedback
Feedback Generation

Correction rules associated with Feedback Template

Extract synthesizer choices to fill templates
Evaluation
Autograder Tool for Python

Currently supports:

- Integers, Bool, Strings, Lists, Dictionary, Tuples

- Closures, limited higher-order fn, list comprehensions
Benchmarks

Exercises from first five weeks of 6.00x and 6.00

**int:** prodBySum, compBal, iterPower, recurPower, iterGCD

**tuple:** oddTuple

**list:** compDeriv, evalPoly

**string:** hangman1, hangman2

**arrays(C#):** APCS dynamic programming (**Pex4Fun**)
<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Test Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>evalPoly-6.00</td>
<td>13</td>
</tr>
<tr>
<td>compBal-stdin-6.00</td>
<td>52</td>
</tr>
<tr>
<td>compDeriv-6.00</td>
<td>103</td>
</tr>
<tr>
<td>hangman2-6.00x</td>
<td>218</td>
</tr>
<tr>
<td>prodBySum-6.00</td>
<td>268</td>
</tr>
<tr>
<td>oddTuples-6.00</td>
<td>344</td>
</tr>
<tr>
<td>hangman1-6.00x</td>
<td>351</td>
</tr>
<tr>
<td>evalPoly-6.00x</td>
<td>541</td>
</tr>
<tr>
<td>compDeriv-6.00x</td>
<td>918</td>
</tr>
<tr>
<td>oddTuples-6.00x</td>
<td>1756</td>
</tr>
<tr>
<td>iterPower-6.00x</td>
<td>2875</td>
</tr>
<tr>
<td>recurPower-6.00x</td>
<td>2938</td>
</tr>
<tr>
<td>iterGCD-6.00x</td>
<td>2988</td>
</tr>
</tbody>
</table>
Average Running Time (in s)
Feedback Generated (Percentage)
Feedback Generated (Percentage)
Feedback Generated (Percentage)

- evalPoly-6.00x
- hangman2-6.00x
- evalPoly-6.00
- hangman1-6.00x
- oddTuples-6.00x
- oddTuples-6.00
- iterPower-6.00x
- iterGCD-6.00x
- recurPower-6.00x
- prodBySum-6.00
- compDeriv-6.00x
- compDeriv-6.00

Feedback Percentage:
- 0.00%
- 10.00%
- 20.00%
- 30.00%
- 40.00%
- 50.00%
- 60.00%
- 70.00%
- 80.00%
- 90.00%
Why low % in some cases?

- Completely Incorrect Solutions
- Unimplemented Python Features
- Timeout
  - comp-bal-6.00
- Big Conceptual errors
Big Error: Misunderstanding APIs

• eval-poly-6.00x

```python
def evaluatePoly(poly, x):
    result = 0
    for i in list(poly):
        result += i * x ** poly.index(i)
    return result
```
Big Error: Misunderstanding Spec

• hangman2-6.00x

def getGuessedWord(secretWord, lettersGuessed):
    for letter in lettersGuessed:
        secretWord = secretWord.replace(letter,'_')
    return secretWord
A technique for automated feedback generation
Error Models, Constraint-based synthesis

Provide a basis for automated feedback for MOOCs

Towards building a Python Tutoring System

Thanks! rishabh@csail.mit.edu