Synthesis for Teaching Introductory Programming

Rishabh Singh
Synthesis For Introductory Programming

Automated Grading and Feedback

Program Synthesis

Variation Synthesis

Problem Generation
Grading Assignments

200 Students
Students get unhappy

"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? ...."
Grading Assignments

200 Students

100,000 Students
Autograder

[PLDI 13]
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):
    length = int(len(poly)-1)
    i = length
    deriv = range(1,length)
    if len(poly) == 1:
        deriv = [0.0]
    else:
        while i >= 0:
            new = round((poly[i] * i),2)
            i -= 1
            deriv[i] = new
    return deriv

Change to length + 1

Change to >
Overview of Autograder Algorithm

> 30 years

10^12 variants

10^6 inputs

Teacher’s solution

Student’s solution

x = x + y

z = x - y

z = x / y

z = x + 1

z = x + y

x = x + y
# Benchmark Set: Thousands of Incorrect Attempts

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Incorrect Attempts</th>
</tr>
</thead>
<tbody>
<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>
100,000 assignments graded for $13.9 on Amazon EC2
# Average Performance

<table>
<thead>
<tr>
<th>TestSet</th>
<th>Generated Feedback</th>
<th>Percentage</th>
<th>AvgTime(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>13365</td>
<td>8579</td>
<td>64.19%</td>
<td>9.91</td>
</tr>
</tbody>
</table>
Autograder For Other Languages

- C
- C++
- Java
- Python
- Scala
- OCaml
- C#
- JSim
Classroom (6.00) deployment

Worked well for week 2 & 3 😊
161/167 (eval-poly), 135/155 (compute-root)

TA: “I didn't even realize that this was a common mistake...”

compute-interest 😞 (float, 16000 iterations)

Concolic Synthesis
edX (6.00x) deployment

A/B test study planned for edX
Semantic Clustering of Student Solutions [Learning@Scale 2014]
def getGuessedWord(secretWord, lettersGuessed):
    ""
    secretWord: string, the word the user is guessing
    lettersGuessed: list, what letters have been guessed so far
    returns: string, comprised of letters and underscores that represents
             what letters in secretWord have been guessed so far.
    ""
    output = ()
st = ""
output = ""
output = []
for char in secretWord:
    if lettersGuessed.count(char) == 0:
        output = output + ('_',)
    else:
        output = output + (char,)
    result += char
else:
    result += '_'
return st.join(output)
return output
for char in lettersGuessed:
    if char not in lettersGuessed:
        result += ' '
    else:
        result += char
        stringList.append(char)
    else:
        stringList.append('_ _')
return ''.join(stringList)
def getGuessedWord(secretWord, lettersGuessed):
    
    secretWord: string, the word the user is guessing
    lettersGuessed: list, what letters have been guessed so far
    returns: string, comprised of letters and underscores that represents
    what letters in secretWord have been guessed so far.
    
    # Initialize variables
    output = []
    st = ""
    output = []

    # Iterate through each character in the secret word
    for char in secretWord:
        if char not in lettersGuessed:
            result += '_'
            else:
                result += char
        else:
            result+=char
    return st.join(output)

    # Iterate through each character in the secret word
    return output

# Example usage
secretWord = "hello"
lettersGuessed = ['e', 'l', 'o']
getGuessedWord(secretWord, lettersGuessed)
Conclusion

Automated Grading and Feedback

Program Synthesis

Variation Synthesis

Problem Generation

rishabh@csail.mit.edu