Synthesis for Automated Grading and Feedback

EXCAPE NSF SITE VISIT AUGUST 2013
Leverage Synthesis in Education Technology

- Computer aided approach to introduce automation in education
Education Technology in ExCape

Focus on 2 problem domains
- Introductory programming
- Automata theory

Key application areas for synthesis
- Problem Generation
- Automated Grading
- Feedback
The challenge
• 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 (>10k students)

Bigger Challenge in MOOCs
AutoGrader

AUTOMATED GRADING FOR PROGRAMMING ASSIGNMENTS

STUDENT: RISHABH SINGH AT MIT
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

replace derive by [0]
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
Solution Idea

Constraint-based Synthesis

Problems satisfy minimum changes.

All Programs

candidates

$10^{15}$
Autograder Algorithm
Algorithm
Algorithm: Rewriter
Simplified Error Model

- \( \text{return } a \rightarrow \text{return } \{[0],\, ?a\} \)
- \( \text{range}(a_1, \ a_2) \rightarrow \text{range}(a_1+1, \ a_2) \)
- \( a \rightarrow a + 1 \)
Rewriting using Error Model

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

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

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

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

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

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

range(0, len(poly))

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

\( a \rightarrow a+1 \)
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, 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

*.py* → *.sk*
Sketch [Solar-Lezama et al. ASPLOS06]

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

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

Statically typed C-like language with holes
Translation to Sketch

(1) Handling python’s dynamic types

(2) Translation of expression choices
Algorithm: Solver
Algorithm: Feedback

Rewriter → Translator → Solver → Feedback

.out
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)
Why low % in some cases?

• Completely Incorrect Solutions
• Unimplemented Python Features
• Timeout
  • comp-bal-6.00
• Big Conceptual errors
AutomataTutor

STUDENTS AND POSTDOCS: LORIS D’ANTONI (PENN)
DILEEP KINI (UIUC), MAHESH VISWANATHAN (UIUC)
Twice ab

Draw a DFA that accepts the following language over the alphabet \( \{a, b\} \): all strings in which 'ab' appears exactly twice as a substring.

Answer:
Automata Feedback Example

The student misunderstood the problem at least 2 occurrences of ‘ab’ instead of exactly 2 occurrences of ‘ab’.

Tool answer

- **Feedback:** The correct language is \{ s \mid ‘ab’ appears in s exactly 2 times \}
- **Grade:** 5/10
Big Error: Misunderstanding Spec

- hangman2-6.00x

```python
def getGuessedWord(secretWord, lettersGuessed):
    for letter in lettersGuessed:
        secretWord = secretWord.replace(letter, '_')
    return secretWord
```
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
```
PLOOC Workshop

PL technology for Open Online Courses

Co-located with PLDI

- ~25 attendees