Automating Generation of Programming Problems

Amy Chou and Justin Kaashoek
Mentor: Rohit Singh
High Level Overview

• **Goal**: Create a system to *automatically generate Python programming problems* with characteristics specified by users

• **Purpose**: Help teachers and students get personalized problems (classroom or MOOC setting)

• **Strategy**: Synthesize a model of a Python AST (Abstract Syntax Tree) using Sketch
Example of a Programming Problem
Example of a Programming Problem

I want to practice these constructs: 
arithmetic, recursion, if-then-else statements
Ok! Here is a problem for you. Fill in the blanks:

```c
int sumDigits(int x){
    if(x/10 != 0){
        return ____ + sumDigits(___);
    } else return x;
}
```
Example of a Programming Problem

I’ll give you some hints.

Input: 123  Output: 6
Input: 444  Output: 12
Input: 12   Output: 3
Example of a Programming Problem

Here is my solution:

```c
int sumDigits(int x){
    if(x/10 != 0){
        return x%10 + sumDigits(x/10);
    } else return x;
}
```
Preliminaries
Python Abstract Syntax Trees (ASTs)

\[
x = 1 \\
y = 2 \\
3 \times (x+y)
\]
Sketch Program Synthesis

Sketch Specification

```c
harness void doublesketch(int x){
    int t = x * ??;
    assert t == x + x;
}
```

- ?? : arbitrary instances of expressions/statements
- Full fledged C++ code in the spec
Sketch Program Synthesis

Sketch Solution

```c
harness void doublesketch(int x){
    int t = x * 2;
    assert t == x + x;
}
```

- ?? : arbitrary instances of expressions/statements
- Full fledged C++ code in the spec
Details
System Overview

User gives constructs → Translation → Sketch Specification → Sketch Solving → Python AST + I/O examples

User Interface
- Input
- Check
- Feedback

Presentation → Programming Problem

Abstraction
ASTs in Sketch: Algebraic Data Types (ADTs)

Constructs:

```
adt expr{
}
int interpret(expr e, int[] context){
    switch(e){
    ...
```
ASTs in Sketch: Algebraic Data Types (ADTs)

Sketch Specification

```plaintext
adt expr{
    Num {int val;}
}
```

```plaintext
int interpret(expr e, int[] context){
    switch(e){
        case Num: return e.val;
    }
}
```

Constructs:

integers
ASTs in Sketch: Algebraic Data Types (ADTs)

Constructs:
- integers
- variables

Sketch Specification

def adt expr{
    Num {int val;}
    Var {int id;}
}

def interpret(expr e, int[] context){
    switch(e){
        case Num: return e.val;
        case Var: return context[var.id];
    }
}

...
ASTs in Sketch: Algebraic Data Types (ADTs)

Sketch Specification

```java
adt expr{
    Num {int val;}
    Var {int id;}
    Plus {expr left; expr right;}
    Mult {expr left; expr right;}
}

int interpret(expr e, int[] context){
    switch(e){
        case Num: return e.val;
        case Var: return context[var.id];
        case Plus: {
            int left = interpret(e.left, context);
            int right = interpret(e.right, context);
            return left + right;
        }
    }
    ...
```
ASTs in Sketch: Algebraic Data Types (ADTs)

Constructs:
- integers
- variables
- arithmetic
- recursion
- function calls
- lists
- assignments
- if-then-else
- while loops
- for loops

Sketch Specification

```
adt expr{
    Num {int val;}
    Var {int id;}
    Plus {expr left; expr right;}
    Mult {expr left; expr right;}
    ...
}
int interpret(expr e, int[] context){
    switch(e){
        case Num: return e.val;
        case Var: return context[var.id];
        case Plus: {
            int left = interpret(e.left, context);
            int right = interpret(e.right, context);
            return left + right;  }
        ...
    }
    ...
```

ASTs in Sketch: Algebraic Data Types (ADTs)
System Overview

User gives constructs → Translation → Sketch Specification

Sketch Solving

Python AST + I/O examples

Presentation

Programming Problem

Abstraction

User Interface
- Input
- Check
- Feedback
Synthesis Specification

```c
adt expr{
    Num {int val;}...
}
int interpret(expr e, int[] context){
    switch(e){
        case Num: return e.val;
        ...
    }
}

//synthesis specification
harness synthesize(){
    expr e = ??(3); //AST of depth 3
    int[] inps = ??; int outp = ??; //input-output example
    assert(interpret(e, inps)) == outp);
}
```
Synthesized AST with I/O Examples

Program
If (a<b):
  b + c
else:
  a + c

Input/Output Examples
(0, 0, 0) → (0)
(1, 2, 3) → (5)
(-3, 2, -1) → (1)
(5, 3, -2) → (3)
Synthesis: Challenges

• Finding ASTs that are **not trivially reducible**
  – e.g. \((a+0)\) or \((a*1)\) or if(b) then a else a
  – Adding “tainted” values to interpret function
    • Making sure that each top level node (input or constant) taints the output
  – Searching for many I/O examples (5 by default)

• **Scalability of Synthesis**
  – Parameters to control search space
    • Depth of AST
    • Number of each type of node to search for
System Overview

User gives constructs → Translation → Sketch Specification → Sketch Solving → Python AST + I/O examples

User Interface:
- Input
- Check
- Feedback

Presentation → Programming Problem → Abstraction
Abstracted Programming Problem

If

<

a b

+

a c

+

b c
Abstracted Programming Problem
Abstracted Programming Problem

Programming Problem

If (a ?? b):
  b + ??
else:
  ?? + c

PRIMES Conference May 17, 2015
System Overview

User gives constructs → Translation → Sketch Specification → Sketch Solving

Python AST + I/O examples → Abstraction

User Interface
- Input
- Check
- Feedback

Programming Problem → Presentation

PRIMES Conference May 17, 2015
Problem Presentation

User sees:
If (a __ b):
   b + __
else:
   __ + c

Input/Output Examples
(0, 0, 0) → (0)
(1, 2, 3) → (5)
(-3, 2, -1) → (1)
(5, 3, -2) → (3)
Problem Presentation

User’s attempt:
If (a != b):
  b + c
else:
  a + c

Input/Output Examples - Feedback
(0, 0, 0) → (0)
(1, 2, 3) → (5)
(-3, 2, -1) → (1)
(5, 3, -2) → (3)
Problem Presentation

User’s second attempt:
If (a < b):
   b + c
else:
   a + c

Input/Output Examples - Satisfied!
(0, 0, 0) → (0)
(1, 2, 3) → (5)
(-3, 2, -1) → (1)
(5, 3, -2) → (3)
Results
ASTs/Programs generated by Sketch

Program:
If (c < b):
    b + b
else:
    a + b

Input/Output Examples
(0,2,1) → 4
(0,5,3) → 10
(30,1,2) → 31
(0,1,0) → 2
(7,5,3) → 10
Program:
If (c < b):
  a * a
else:
  b + a

Input/Output Examples
(2,1,5) → 6
(5,1,0) → 25
(3,2,0) → 9
(3,8,0) → 9
(1,4,6) → 10
Program:
If (b < c):
    a * c
else:
    a * b

Input/Output Examples
(2,1,3) → 6
(3,3,2) → 9
(6,4,5) → 30
(4,1,7) → 28
(10,2,1) → 20
Program:
If (c < a):
    2*c + b
else:
    2*c

Input/Output Examples
(4,26,0) → 26
(7,11,10) → 20
(0,11,15) → 30
(8,16,12) → 24
(30,28,1) → 30
Program:
If (c > 3):
  2
else:
  a+c

Input/Output Examples
(0,24,4) → 2
(0,0,4) → 2
(0,8,6) → 2
(8,16,8) → 2
(8,0,1) → 9
Program:
If (c < 4):
a+b
else:
a+c

Input/Output Examples
(4,2,3) → 6
(2,3,0) → 5
(4,0,8) → 12
(3,3,0) → 6
(1,0,10) → 11
Current Status and Future Work

• Current Status
  – Parts of the pipeline Independently tested
    • Python ASTs
    • Sketch Synthesis
  – Working on putting them together

• Future work/Improvements
  – Automating different processes
    • Abstraction with heuristics
    • Generation of Sketch Specifications
    • Generating optimal problems
  – UI (input, verify, feedback)
Acknowledgements

We would like to thank...

- Our mentor Rohit Singh for his guidance and insights
- Professor Armando Solar-Lezama for suggesting the problems and his guidance
- MIT PRIMES faculty
- Our parents!