QLOSE: Program Repair with Quantitative Objectives

Loris D'Antoni

WISCONSIN
UNIVERSITY OF WISCONSIN-MADISON

Roopsha Samanta

PURDUE
UNIVERSITY

Rishabh Singh

Microsoft Research

Everyone wants to be a programmer

Programming for Everybody (Getting Started with Python)

UNIVERSITY OF MICHIGAN

657,068

An Introduction to Interactive Programming in Python

RICE

581,043

Introduction to Computer Science

UDACITY

515,476

Learn to Programming: The Fundamentals

UNIVERSITY OF TORONTO

198,566

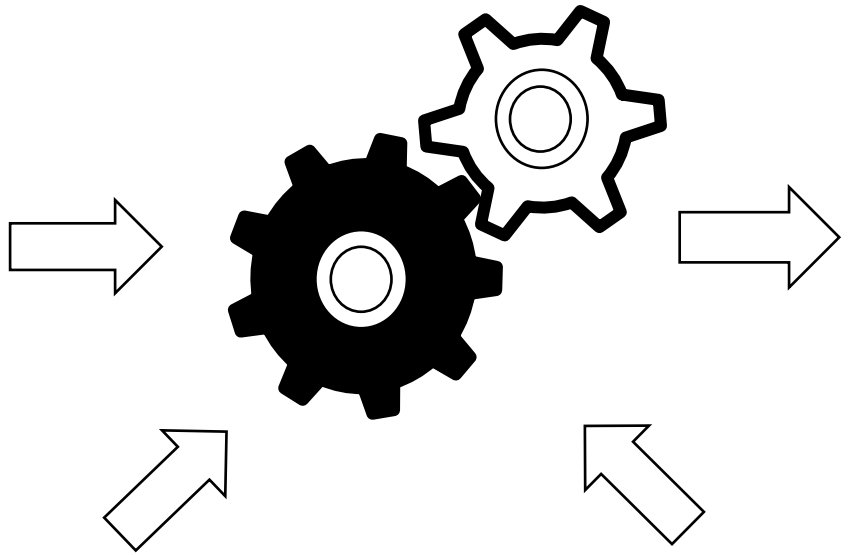
Introduction to Computer Science and Programming

Massachusetts Institute of Technology

157,431

Basic strategy for automated feedback/grading

```
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]

replace by 1

replace by false

Teacher's Solution

Error Model

[R. Singh, S. Gulwani, A. Solar-Lezama. Automatic Feedback Generation for Introductory Programming Assignments. PLDI, 2013]

Input			Output	
s	**c**	**k**	**expected**	**actual**
ab?	?	2	true	**false**
aba?gc	?	5	true	**true**
?aba	?	3	true	**true**

(Return true if character c occurs in string s before position k+1)

```
Bad(str s, ch c, int k){
  for(int j=0;j<k;j++)
     if(c==’?’)
        return true;
   return false;
 }
```

```
Find(str s, ch c, int k){
  for(int j=0;j<k;j++)
     if(s[j]==c)
        return true;
   return false;
 }
```

```
Good(str s, ch c, int k){
  for(int j=0;j<=k;j++)
     if(s[j]==c)
        return true;
   return false;
 }
```

Can we formalize why Good is better?

Quantitatively Close

QLOSE: Program Repair with Quantitative Objectives based on Program Distances

Syntactic distances

- *How much did the repair change the program syntactically?*
  - No. of expression modifications
  - Size of a modified expression
  - …

```
Find(str s, ch c, int k){
  for(int j=0;j<k;j++)
     if(s[j]==c)
        return true;
   return false;
}
```

```
Bad(str s, ch c, int k){
  for(int j=0;j<k;j++)
     if(c==’?’)
        return true;
   return false;
 }
```

Syntactic distances

- *How much did the repair change the program syntactically?*
  - No. of expression modifications
  - Size of a modified expression
  - …

```
Find(str s, ch c, int k){
  for(int j=0;j<k;j++)
     if(s[j]==c)
        return true;
   return false;
}
```

```
Good(str s, ch c, int k){
  for(int j=0;j<=k;j++)
     if(s[j]==c)
        return true;
   return false;
 }
```

Semantic distances

- *How much did the repair change the correct program executions?*
  - Hamming distance between program executions

```
1 Find(str s, ch c, int k){
2   for(int j=0;j<k;j++)
3       if(s[j]==c)
4           return true;
5   return false;
6 }
```

Input			Output	
s	c	k	expected	actual
ab?	?	2	true	false
aba?gc	**?**	**5**	**true**	**true**
?aba	?	3	true	true

s	c	k
aba?gc	?	5

```
1 Find(str s, ch c, int k){
2   for(int j=0;j<k;j++)
3       if(s[j]==c)
4           return true;
5   return false;
6 }
```

Loc	j	out
2	⊥	⊥
3	0	⊥
2	0	⊥
3	1	⊥
2	1	⊥
3	2	⊥
2	2	⊥
3	3	⊥
4	3	⊥
6	⊥	true

s	c	k
aba?gc	?	5

```
1 Bad(str s, ch c, int k){
2   for(int j=0;j<k;j++)
3       if(c==‘?’)
4           return true;
5   return false;
6 }
```

Loc	j	out
2	⊥	⊥
3	0	⊥
4	0	⊥
6	⊥	true

Loc	j	out
2	⊥	⊥
3	0	⊥
2	0	⊥
3	1	⊥
2	1	⊥
3	2	⊥
2	2	⊥
3	3	⊥
4	3	⊥
6	⊥	true

s	c	k
aba?gc	?	5

```
1 Good(str s, ch c, int k){
2   for(int j=0;j<=k;j++)
3       if(s[j]==c)
4           return true;
5   return false;
6 }
```

Loc	j	out
2	⊥	⊥
3	0	⊥
2	0	⊥
3	1	⊥
2	1	⊥
3	2	⊥
2	2	⊥
3	3	⊥
4	3	⊥
6	⊥	true

Loc	j	out
2	⊥	⊥
3	0	⊥
2	0	⊥
3	1	⊥
2	1	⊥
3	2	⊥
2	2	⊥
3	3	⊥
4	3	⊥
6	⊥	true

Quantitative program repair

Given a program P, set of permissible edits E and test set S such that P does not satisfy S, find P’:

1. P’ satisfies S
2. P’ is obtained from P using E
3. $P' \in \text{argmin}_{P''} (d_{syn}(P, P''))$

Quantitative program repair

Given a program P, set of permissible edits E, test set S, syntactic distances $d^1_{syn}, d^2_{syn}, \ldots$, semantic distances $d^1_{sem}, d^2_{sem}, \ldots$, and objective functions $f_1, \ldots, f_n$ such that P does not satisfy S, find P':

1. P' satisfies S
2. P' is obtained from P using E
3. $\mathrm{P}' \in \mathrm{argmin}_{\mathrm{P}''} \, \mathrm{AGGREGATE}_{1 \le i \le n} \{ f_i(d^1_{syn}(\mathrm{P}, \mathrm{P}''), \cdots, d^1_{sem}(\mathrm{P}, \mathrm{P}'', \mathrm{S}), \cdots) \}$

Pareto, sorted objective, …

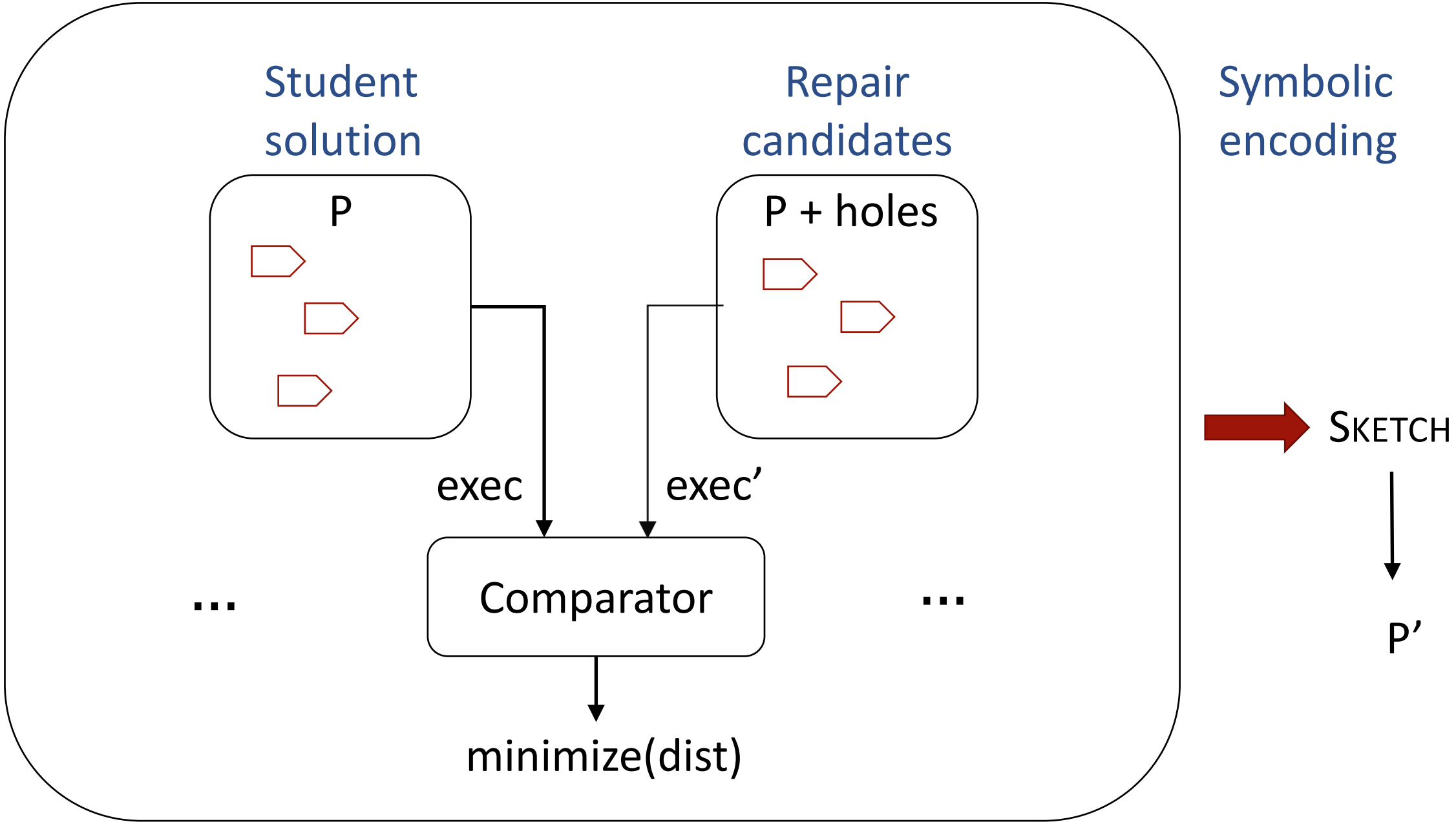
Solution outline

Synthesis + optimization

SKETCH

```
void P(int in){
    int v;
    int y = ?? * in;
    if (in == 10) assert y == 20;
    minimize v;
}
```

[A. Solar-Lezama. Program sketching. STTT, 2013]


Student solution
Repair candidates
Symbolic encoding
P
P + holes
exec
exec'
...
Comparator
...
minimize(dist)
SKETCH
P'

Experiments

Set-up

Benchmarks from CodeHunt, edX, Semfix

Evaluated using 3 objective functions

- Only syntactic distance $(d_{syn}^{\#exp}(\mathrm{P},\mathrm{P}'), d_{syn}^{sizeexp}(\mathrm{P},\mathrm{P}'))$
- Only semantic distance $d_{sem}^{Ham}(\mathrm{P},\mathrm{P}',\mathrm{S})$
- Syntactic + semantic $\left(d_{syn}^{\#exp}(\mathrm{P},\mathrm{P}'), d_{syn}^{sizeexp}(\mathrm{P},\mathrm{P}') + d_{sem}^{Ham}(\mathrm{P},\mathrm{P}',\mathrm{S})\right)$

Good fix: manual inspection

Problem	LOC	|Vars|	|T|	Syntactic		Semantic		Syntactic + Semantic	
FindC	4	4	4	1.5s	❌	2.5s	✅	2.2s	✅
LargestGap-1	10	4	4	9.8s	✅	184.9s	❌	13.4s	✅
LargestGap-2	7	4	4	6.9s	❌	TO	-	18.2s	✅
LargestGap-3	8	4	4	7.3s	✅	15.7s	✅	14.4s	✅
tcas-semifix	10	4	5	12.6s	✅	27.8s	✅	18.4s	✅
max3	5	3	4	1.1s	✅	1.7s	❌	1.9s	✅
iterPower-1	7	3	4	2.3s	❌	10.3s	✅	3.4s	✅
iterPower-2	7	3	4	2.1s	✅	15.2s	❌	2.7s	✅
ePoly-1	8	4	3	1.8s	❌	3.6s	✅	2.5s	✅
ePoly-2	10	4	3	2.4s	✅	4.6s	✅	2.8s	✅
multIA	5	4	3	1.8s	❌	21.5s	❌	2.4s	✅

QLOSE: Quantitative program repair
using syntactic + semantic program distances
for automated grading/feedback.

Next: Program distances for more general specifications
Scalability

Questions?