Autograder

RISHABH SINGH, SUMIT GULWANI, ARMANDO SOLAR-LEZAMA

MIT COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE LABORATORY

Microsoft Research

CSAIL

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

- Manual feedback by TAs
  - Time consuming and error prone

Feedback on Programming Assignments

"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****? ...."*

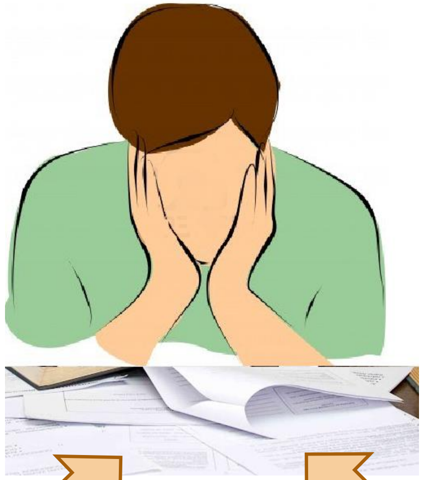
Student Feedback

Scalability Challenges (>100k students)

Bigger Challenge in MOOCs

Today’s Grading Workflow

```
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]

Teacher’s Solution

Grading Rubric

Autograder Workflow

```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
```

replace derive by [0]

Teacher's Solution

Error Model

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

```
poly = [10, 8, 2]   #f(x) = 10 + 8x +2x²
     => [8, 4]      #f'(x) = 8 + 4x
```

Teacher's solution

```python
def computeDeriv(poly):
    result = []
    if len(poly) == 1:
        return [0]
    for i in range(1, len(poly)):
        result += [i * poly[i]]
    return result
```

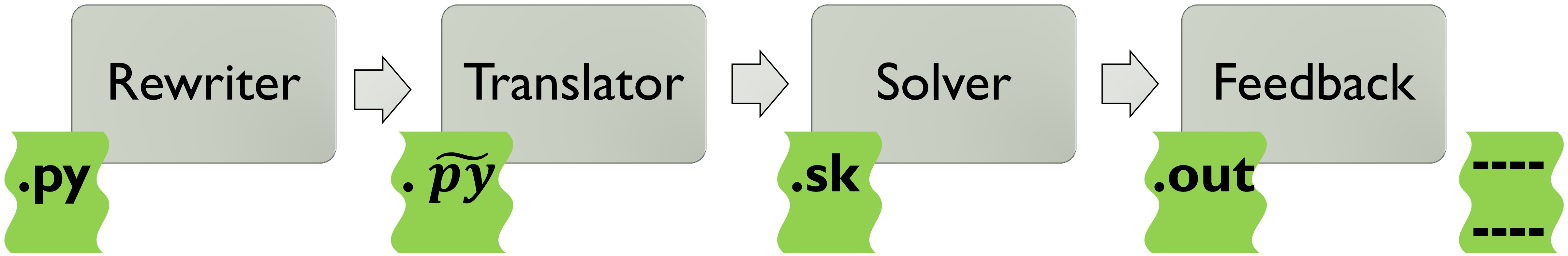
Demo: http://bit.ly/179jFjo

Simplified Error Model

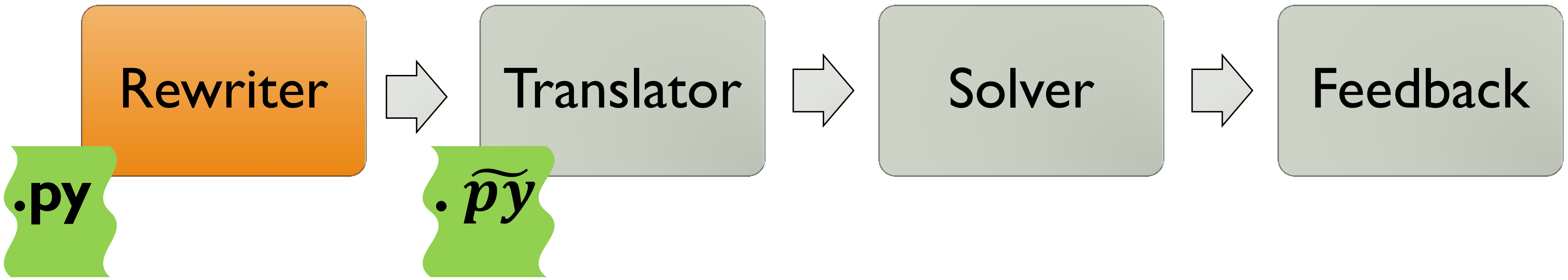
- `return a` → `return {[0],?a}`
- `range(`a_1`, `a_2`)` → `range(`a_1`+1,`a_2`)`
- a_0 `==` a_1 → `False`

Autograder Algorithm

Algorithm


Rewriter
Translator
Solver
Feedback
.py
$.\widetilde{py}$
.sk
.out

Algorithm: Rewriter



Rewriting using Error Model

```
range(0, len(poly))
```

⇓

```
range({0 ,1}, len(poly))
```

default choice

```
a → a+1
```

Rewriting using Error Model

```
range(0, len(poly))
```

```
range({0 ,1}, len(poly))
```

```
a → a+1
```

Rewriting using Error Model

```
range(0, len(poly))
```

```
range({0 ,1}, len({poly, poly+1}))
```

```
a → a+1
```

Rewriting using Error Model

```
range(0, len(poly))
```

```
range({0 ,1}, {len({poly, poly+1}),
len({poly, poly+1})+1})
```

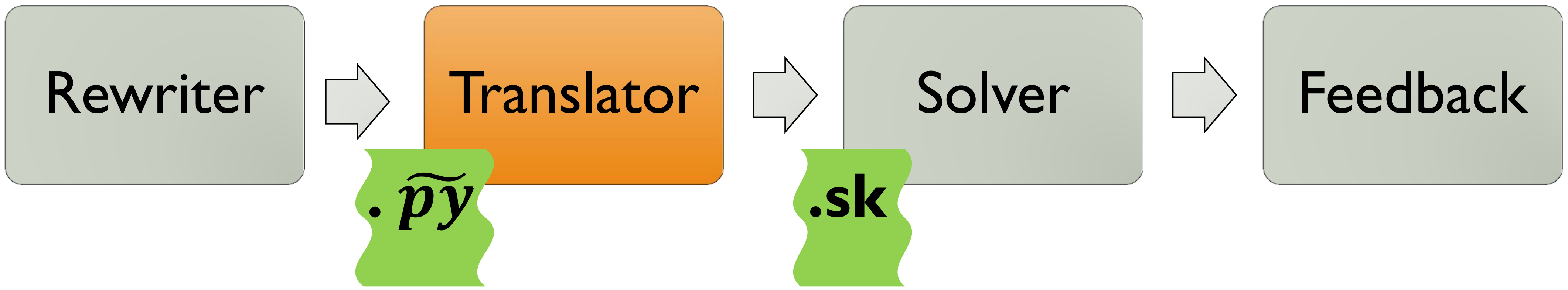
a → a+1

Rewriting using Error Model ($\widetilde{Py}$)

```
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



A Synthesis Primer

The Synthesis problem as a doubly quantified constraint

$$\exists P \; \forall in \quad (in, P \vDash Spec)$$

- 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

$$\exists\, c\ \forall\, in \quad (in, P[c] \vDash Spec)$$

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;
}
```

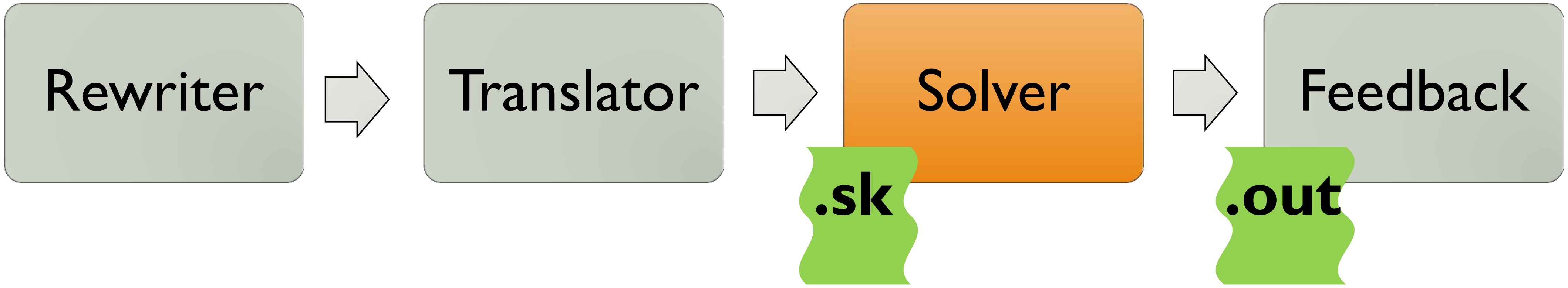
Statically typed C-like language with holes

$\widetilde{Py}$ Translation to Sketch

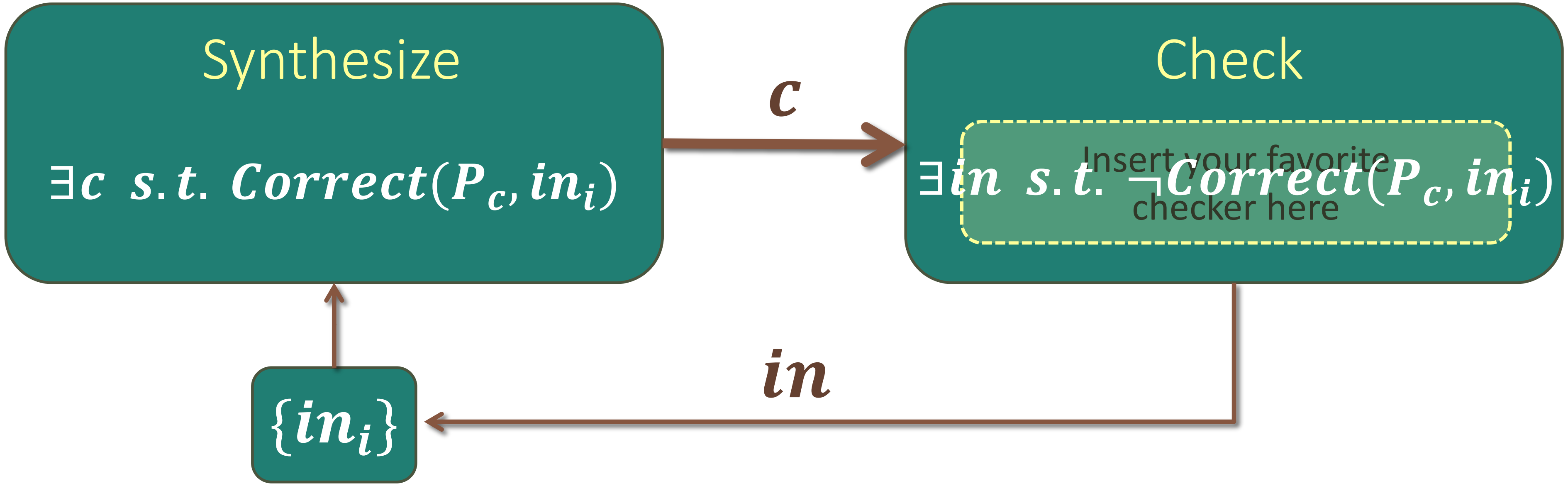
(1) Handling python's dynamic types

(2) Translation of expression choices

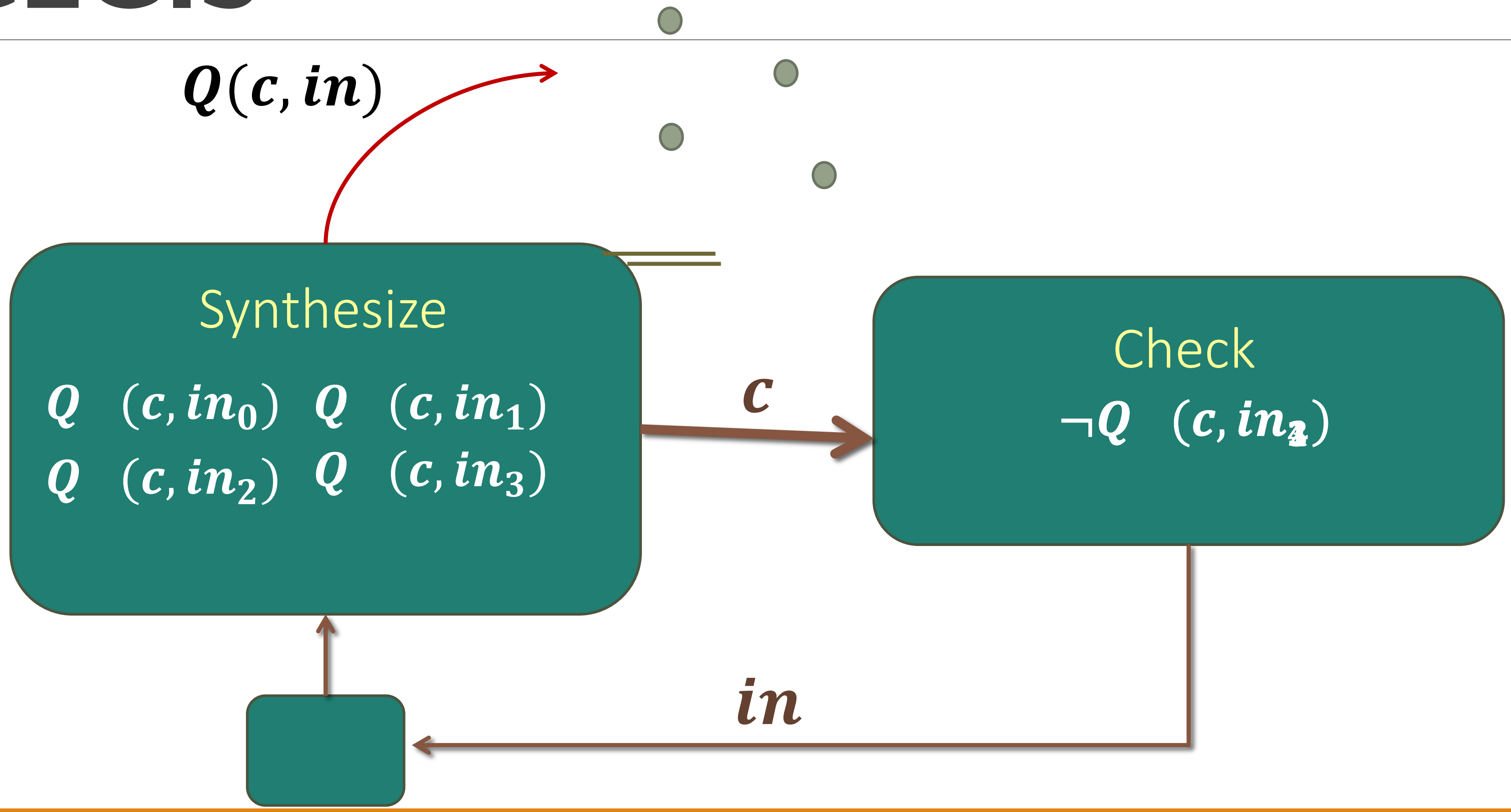
Algorithm: Solver



CEGIS Synthesis algorithm

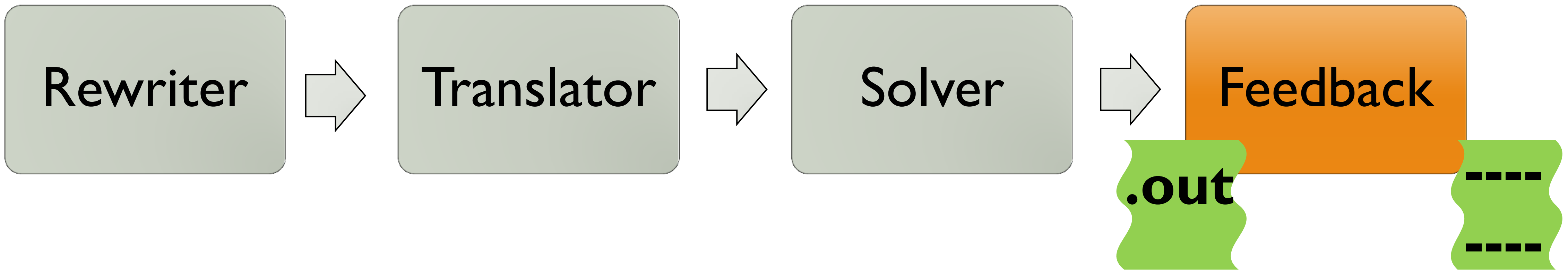


CEGIS



$Q(c, in)$
Synthesize
$Q\ (c, in_0)$ $Q\ (c, in_1)$
$Q\ (c, in_2)$ $Q\ (c, in_3)$
c
Check
$\neg Q\ (c, in_4)$
in

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

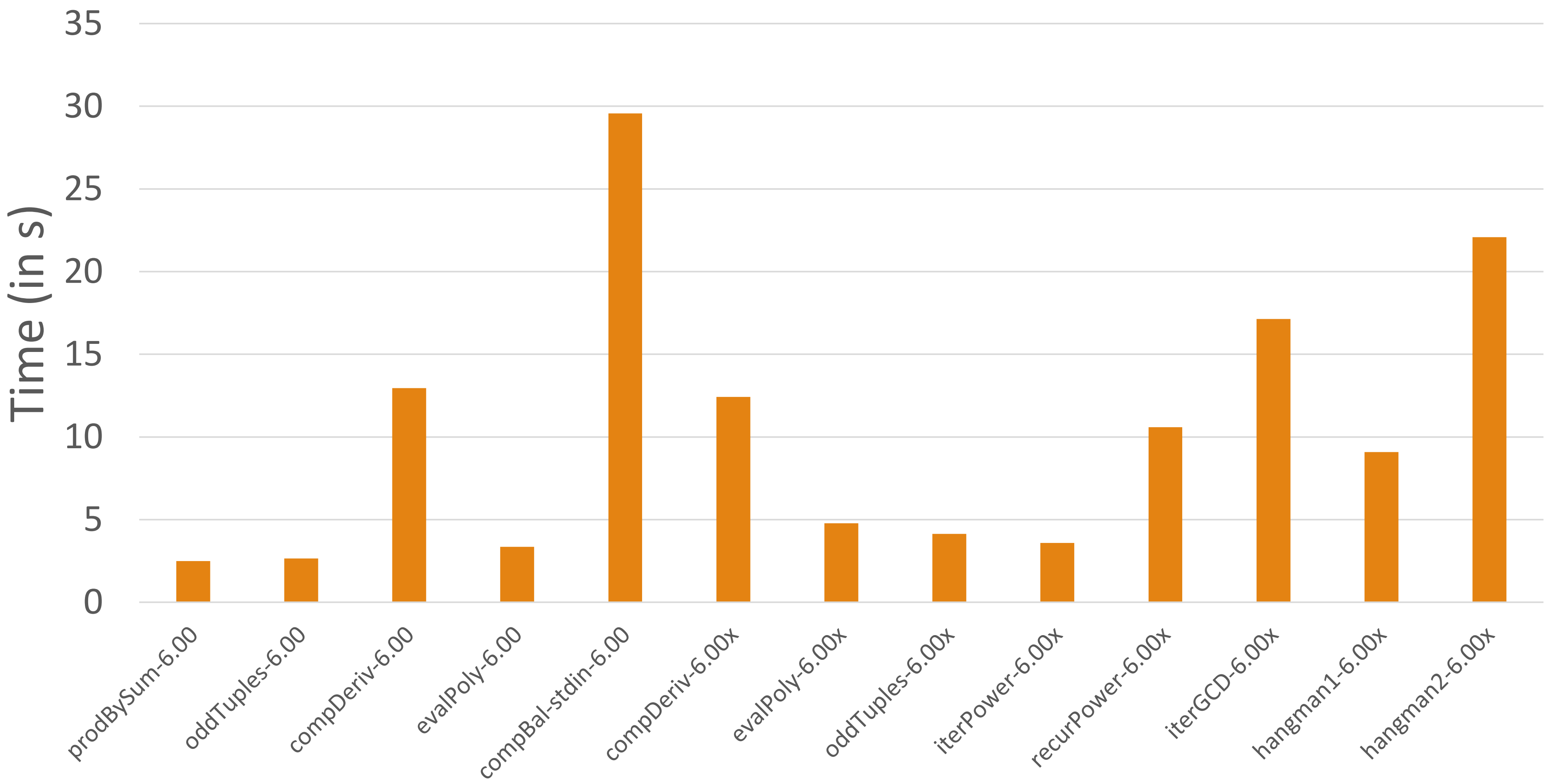
tuple: oddTuple

list: compDeriv, evalPoly

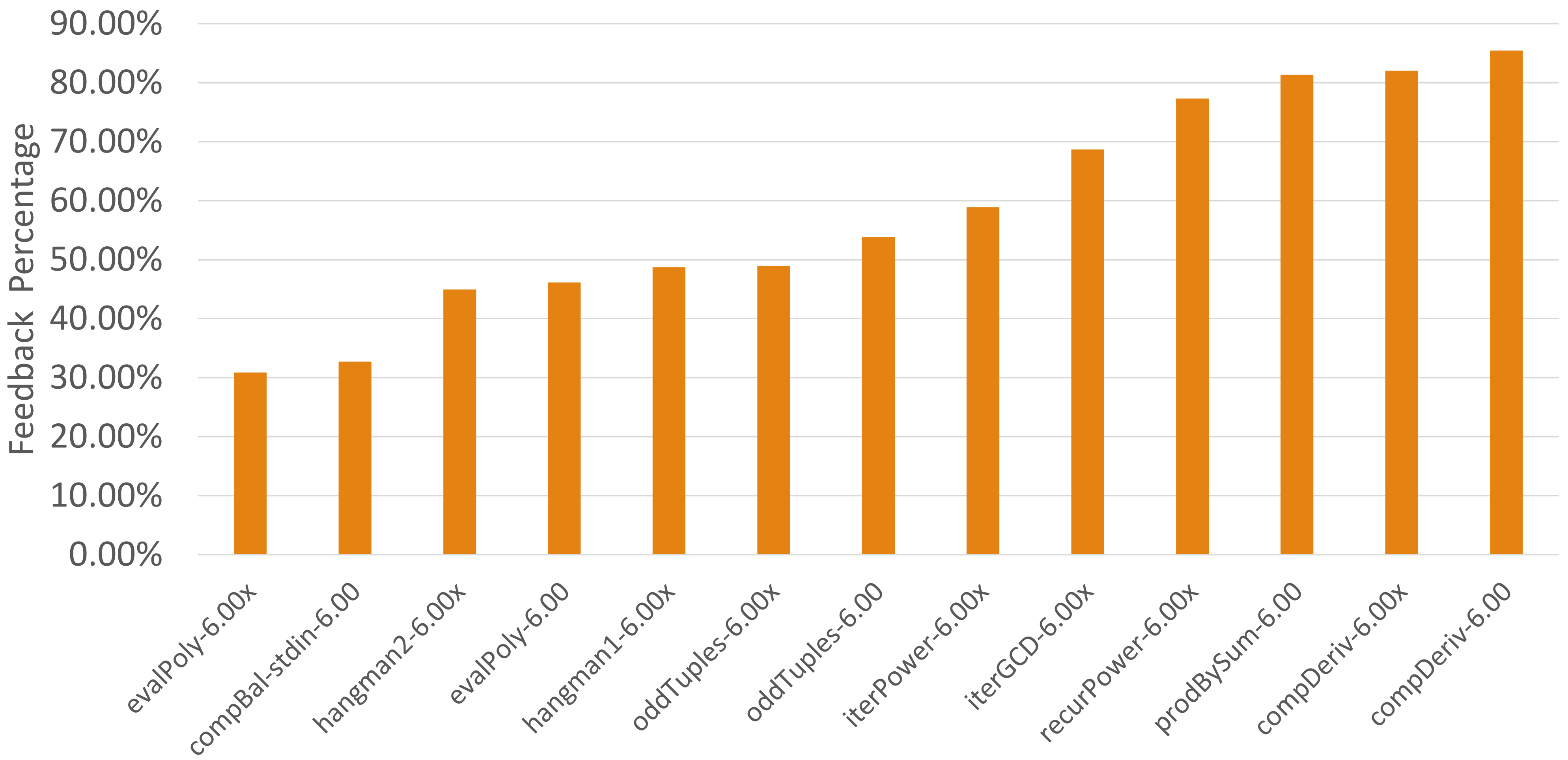
string: hangman1, hangman2

arrays(C#): APCS dynamic programming (**Pex4Fun**)

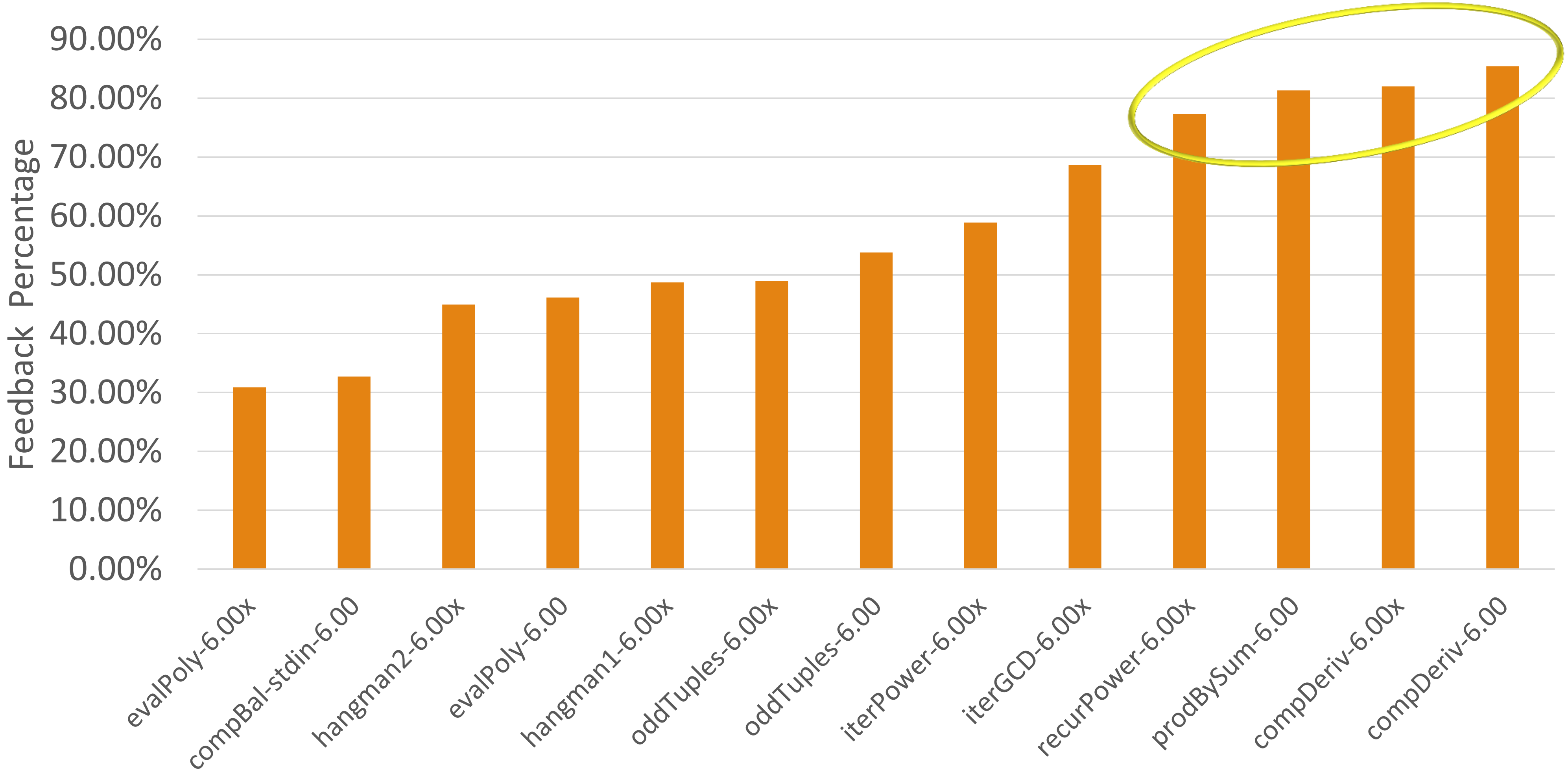
Benchmark	Test Set
evalPoly-6.00	13
compBal-stdin-6.00	52
compDeriv-6.00	103
hangman2-6.00x	218
prodBySum-6.00	268
oddTuples-6.00	344
hangman1-6.00x	351
evalPoly-6.00x	541
compDeriv-6.00x	918
oddTuples-6.00x	1756
iterPower-6.00x	2875
recurPower-6.00x	2938
iterGCD-6.00x	2988



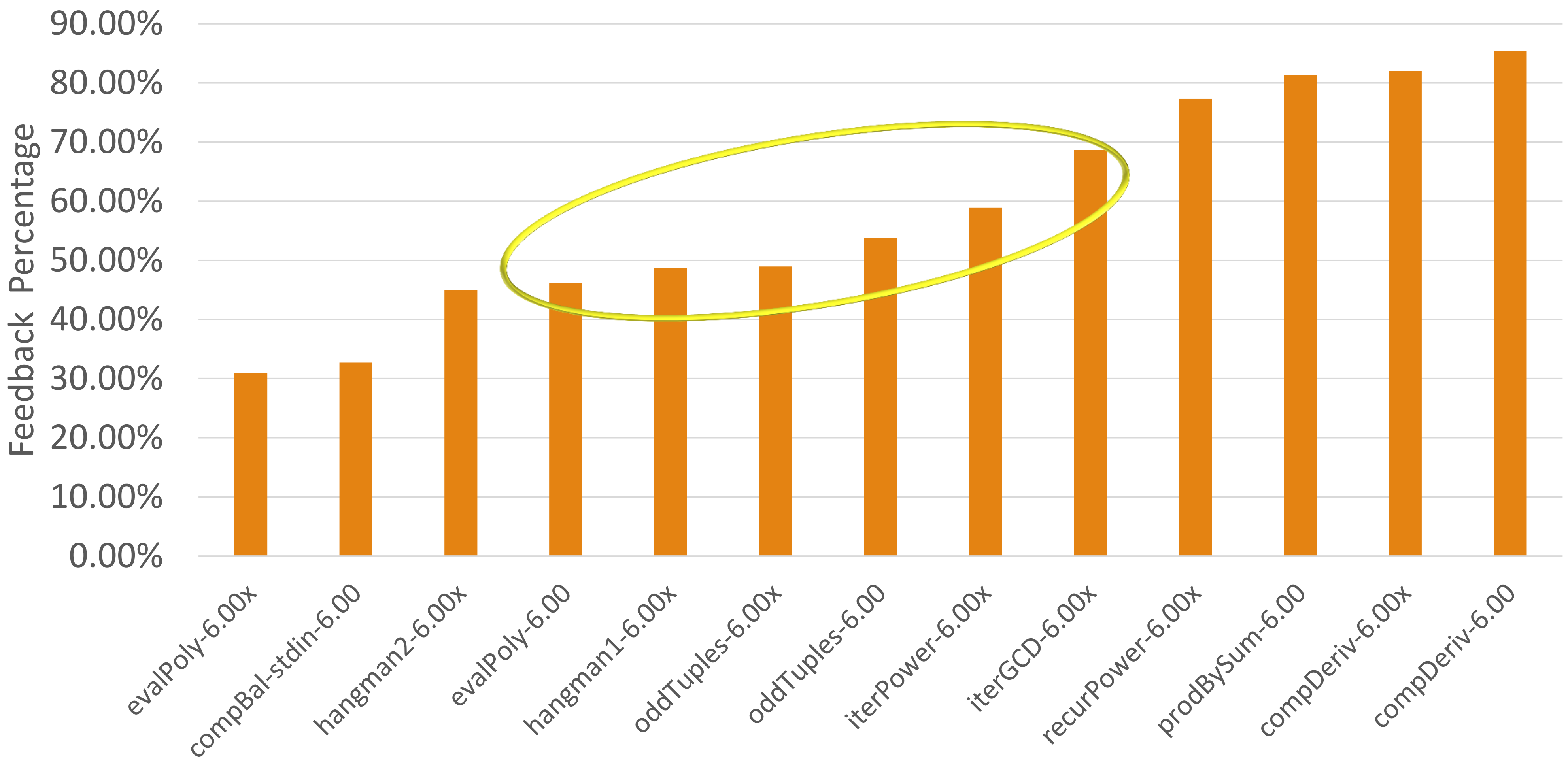
Average Running Time (in s)



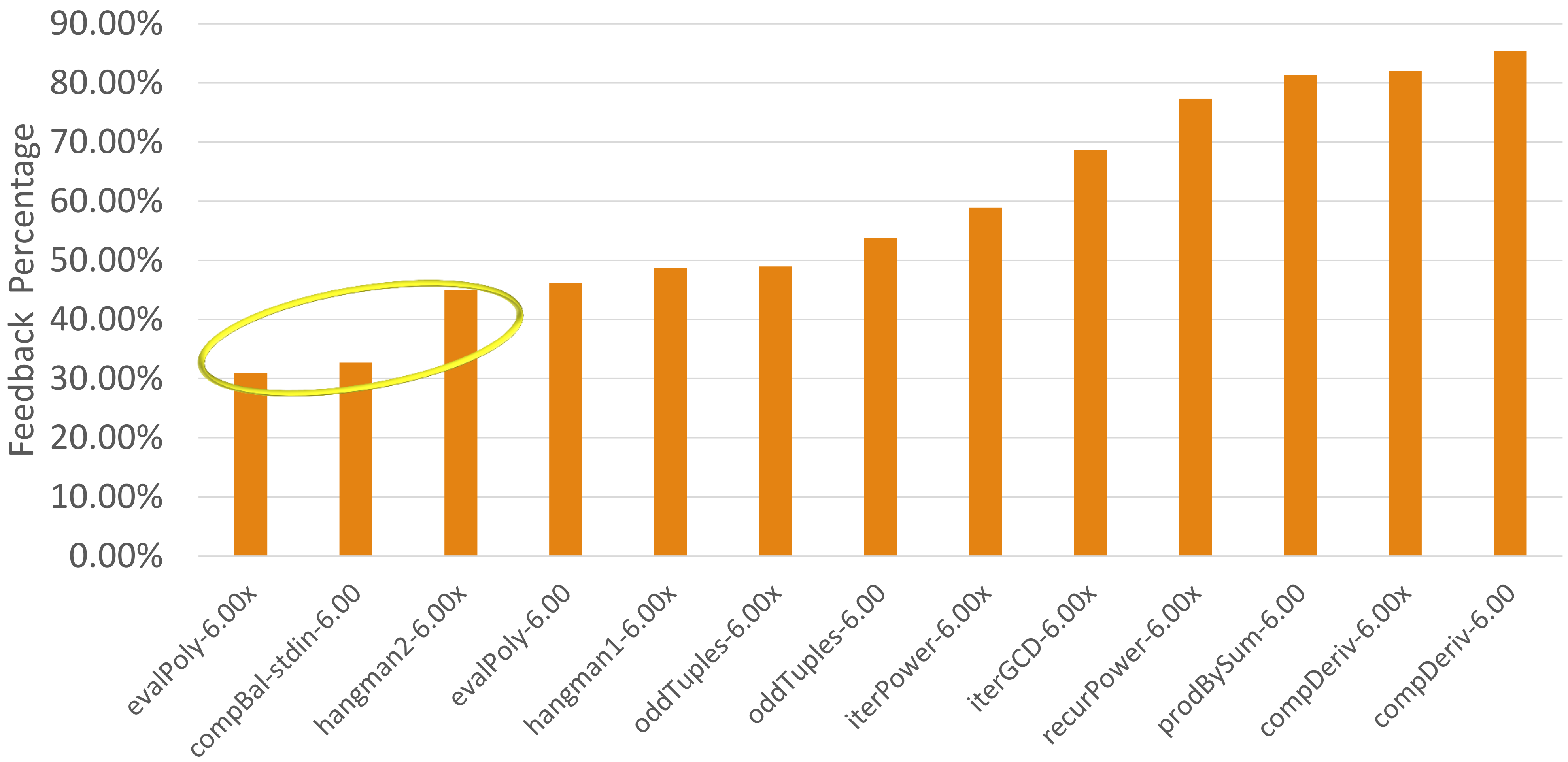
Feedback Generated (Percentage)



Feedback Generated (Percentage)



Feedback Generated (Percentage)



Feedback Generated (Percentage)

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

```
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

Conclusion