Automated Evaluation of Programming Assignments – from Practice to Research

Sujit Kumar Chakrabarti

IIITB

January, 2020

Outline for section 1

Credits

- Nikhila – PhD student
- Ananta – MS/R student
- Kapil Kalra – summer intern, 2018
- Manish Gupta – Professor (IIIT-B), Google AI Research Lab
- Many students, TAs, project/thesis students, interns

Why Automated Evaluation?

- Online learning platforms: Coursera, Udacity, EdX ...
- Online programming contests: ACM ICPC, HakerEarth, HackerRank, CodeChef ...
- Introductory programming courses
- Error prone, labour intensive, repetitive

Why Automated Evaluation?

- Online learning platforms: Coursera, Udacity, EdX ...
- Online programming contests: ACM ICPC, HakerEarth, HackerRank, CodeChef ...
- Introductory programming courses
- Error prone, labour intensive, repetitive – *Boring!*

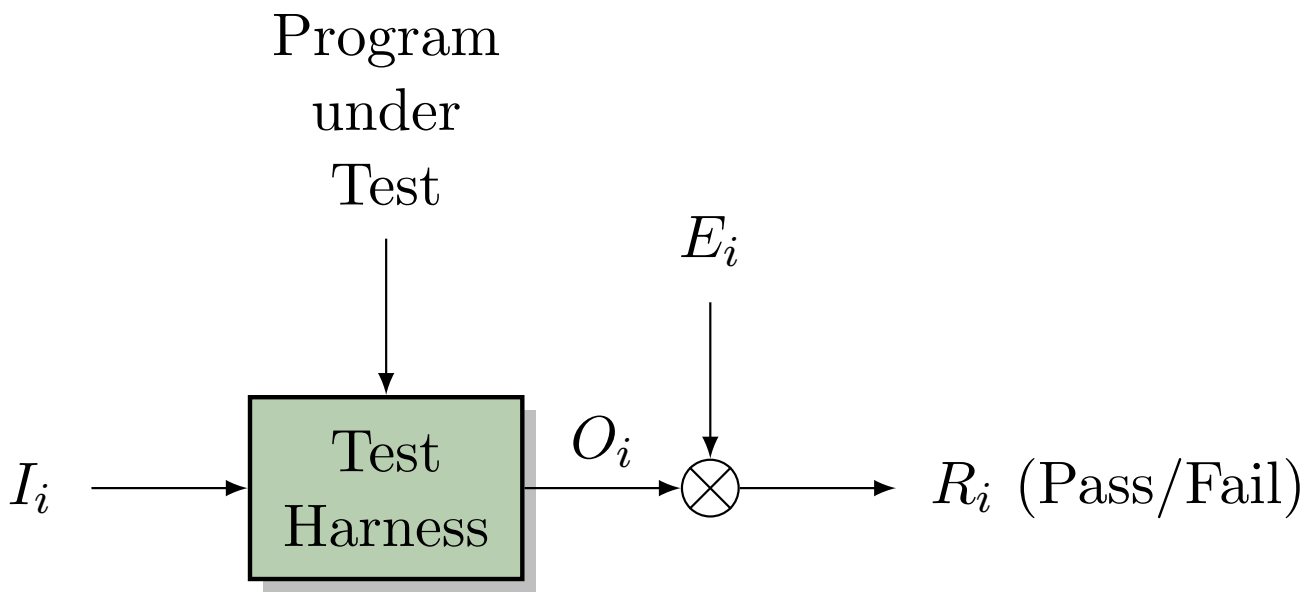
Our Contribution

Automated Evaluation System

- Automatically evaluates programming assignments using testing
- Several human weeks $\longrightarrow$ a few seconds
- Has enabled more frequent, deeper formative assessments with shorter feedback cycles

Testing

A Test Setup



I_i	Test input
E_i	Expected output
O_i	Actual output
R_i	Test result

Assigning Marks:

$$M = \sum k_i R_i$$

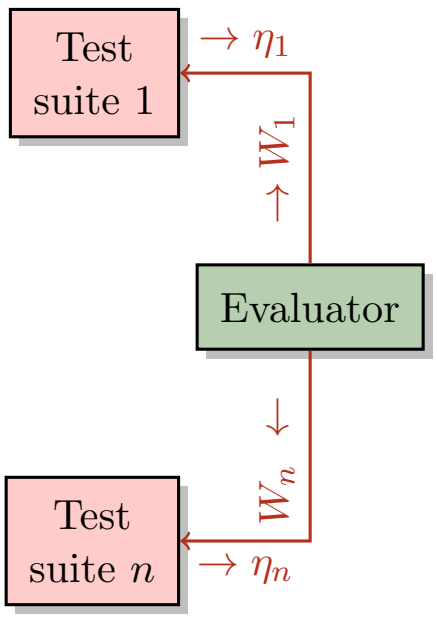
Test Case

Example

```
def eval_triangle_area():

  import mycode.pentagon
  import code.pentagon

  t1 = mycode.pentagon.Triangle(5, 10)
  t2 = code.pentagon.Triangle(5, 10)

  eo = t1.area()
  ao = t2.area()

  if(equals(ao, eo)):
    return (1, "eval_triangle_area")
  else:
    return (0, "eval_triangle_area: wrong answer")
```

Test case to check if a triangle's area is computed properly

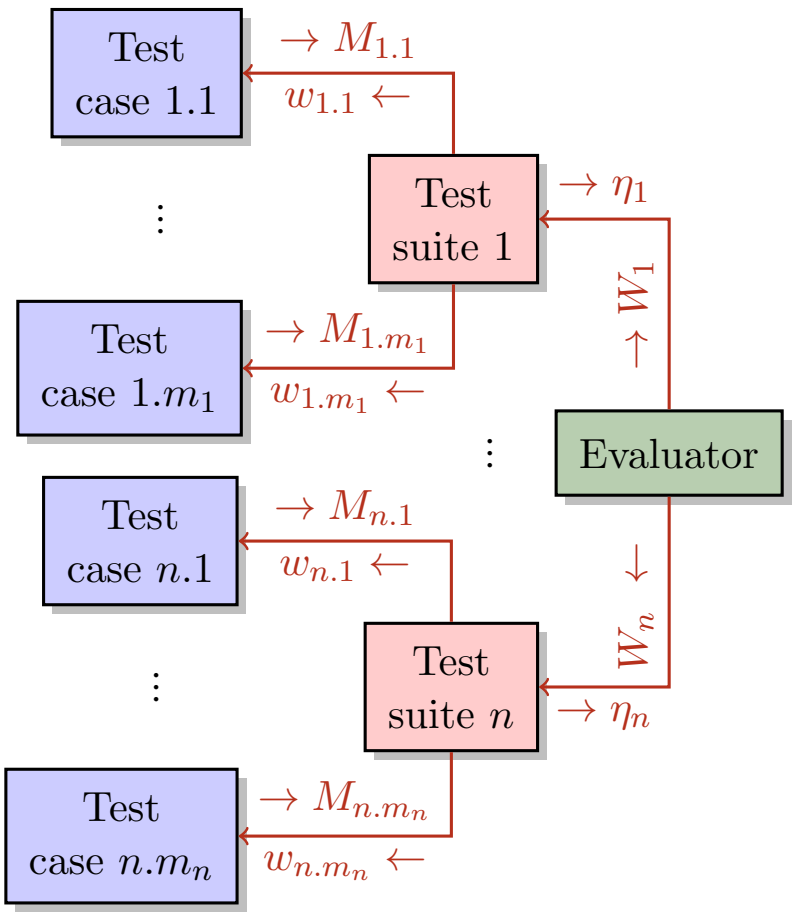
System Architecture



$$\eta = \frac{\Sigma w M}{\Sigma w}$$

$$total = \Sigma \eta W$$

System Architecture



$$\eta = \frac{\Sigma w M}{\Sigma w}$$

$$total = \Sigma \eta W$$

AEPA System

Achievements

1. Simple setup – no server (except LMS)
2. Simple use – on local machine
3. Flexibility
4. Language independent
5. All data is readily available
6. Knowledge creation
7. Software offering

Outline for section 2

Approaches to Automated Evaluation

- Testing
- Testing + Static Analysis
- Static Analysis
- Testing + Static Analysis + Machine Learning

Testing

Advantages:

1. Conceptually simple
2. Easy to automate

Testing

Advantages:

1. Conceptually simple
2. Easy to automate

Issues:

1. Fragility
2. Designing the comparator
3. Testing structural properties

Testing

Issues – Fragility

1. Parsing complex outputs – tolerating minor variability
2. Overhead on students to understand the format and code to print output in a the specified format
3. Crashes and exceptions
4. Infinite loops

Solutions:

1. Running P in a different thread.
2. Wrapping P in input/output script and then executing

Testing

Issues – Testing Structural Properties

Examples:

- Has the factorial function been implemented using recursion or loop?
- Quicksort or mergesort?
- ...

Testing + Static Analysis

Basic Idea:

To use reflection to navigate through the abstract syntax tree of the program?

Example:

```
@evaluate
def eval_square_baseclass():
  import code.pentagon
  s = code.pentagon.Square(10)
  base_names = [c.__name__ for c in s.__class__.__bases__]
  if(base_names == ["RegularPolygon"]):
    return (1, "eval_square_baseclass")
  else:
    return (0, "eval_square_baseclass: wrong answer")
```

Testing + Static Analysis

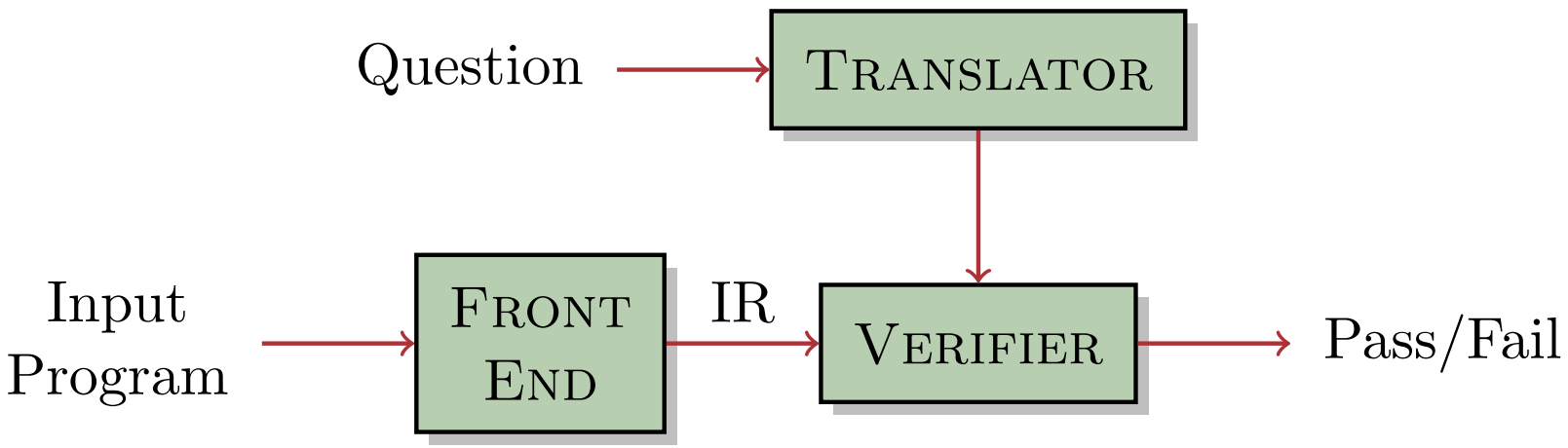
Issues

- The static properties that can be tested is dependent on the depth of reflection offered by the programming language. For example, in C++, how to check if class B is subclass of class A?
- Some properties are not easy to test; require more full-fledged static analysis. For example, how to test if a variable used has been defined earlier or not?

Outline for section 3

Static Analysis

Basic Approach



Static Analysis

Issues – Partially correct submissions

Example

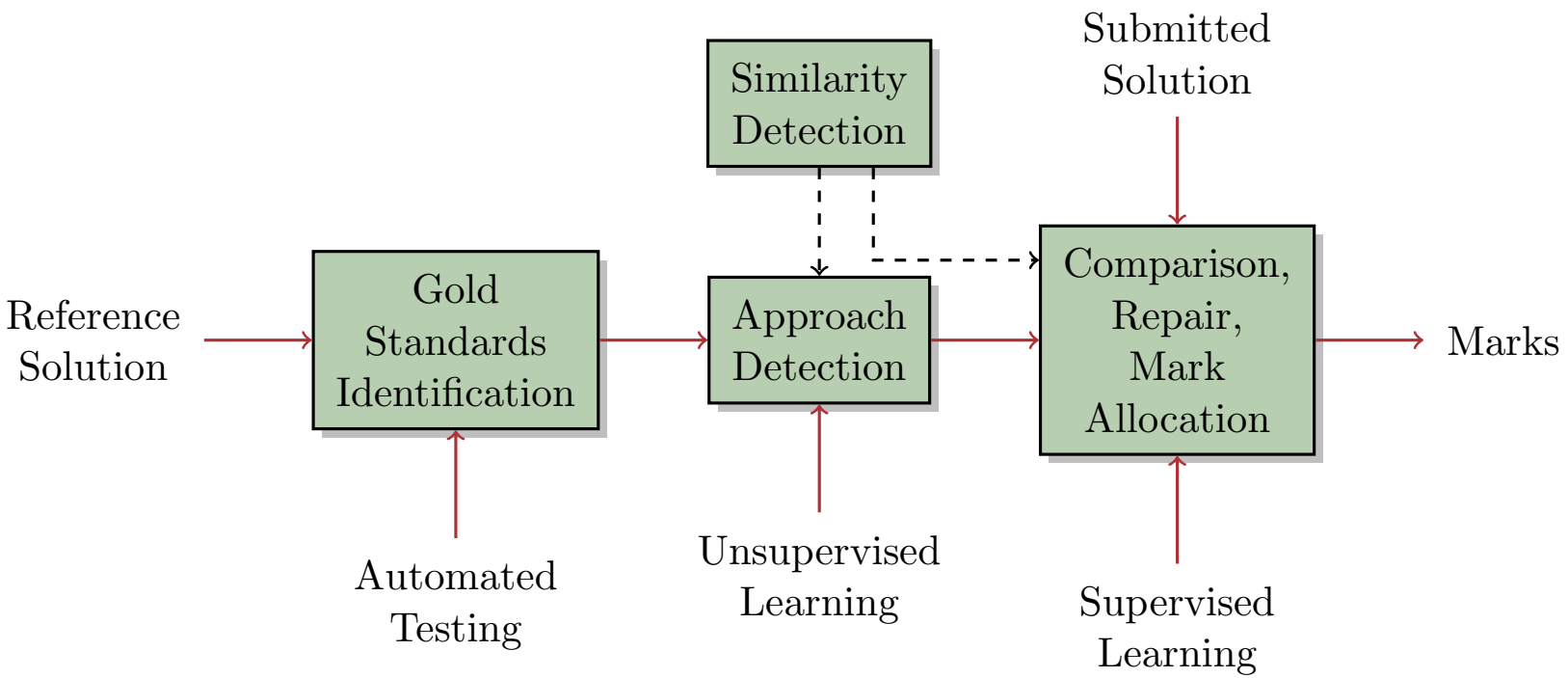
Q. Write a program that finds the transverse of a matrix (represented as a list of lists).

```
m = [  [1, 2, 3],  [4, 5, 6] ]
n = []
for i in range(len(m[0])):
  row = []
  for j in range(len(m)):
    row.append(m[j][i])
  n.append(row)
print n
```

Outline for section 4

Static Analysis + Machine Learning

Basic Approach



Outline for section 5

Conclusions

Summary

- A home-grown automated evaluation system
- Approaches and challenges in automated evaluation
- Static analysis
- Static analysis + machine learning

Thank you!

Thank you!