An Overview on Diversity and Software Testing

Héctor D. Menéndez

Department of Computer Science
University College London

3rd April 2018

Diversity

Diversity and Software Testing

Why does Software Testing need diversity?

What is a proper definition of diversity?

How is diversity applied to Software Testing?

Diversity and Software Testing

Why does Software Testing need diversity?

What is a proper definition of diversity?

How is diversity applied to Software Testing?

Testing strategies

Domain Testing (Inputs)

Coverage Testing (Control-Flow, Data-Flow)

Combination Testing (Inputs)

Adaptive Random Testing (Inputs)

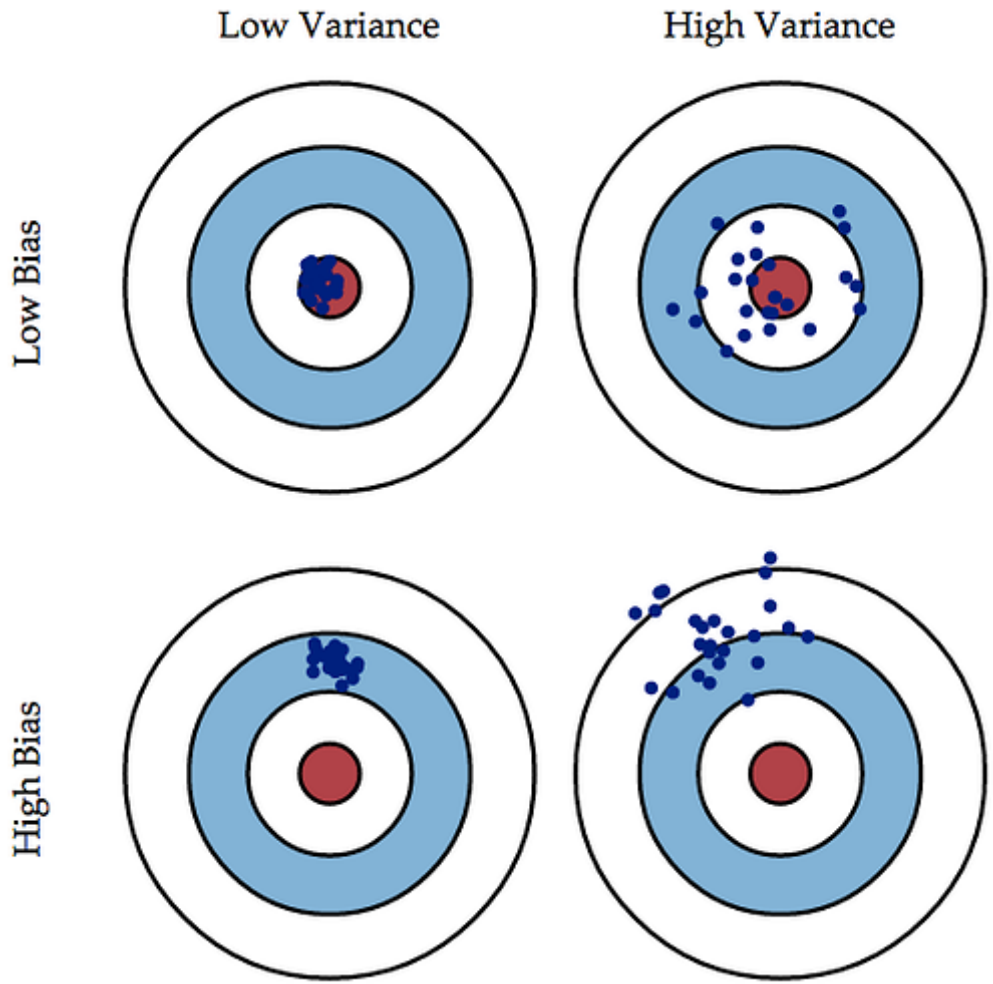
Cluster Test Selection (Behaviour)

Similarity-based Test Selection (Abstraction)

Tests and Bugs

Every test is equally likely to activate or detect a bug

Test Suite Bias and Variance



Test Suite Bias

It is common that the test suite can not generalize the specific problems.

In regression testing some test might finish obsolete.

We need ways to extend the tests suites.

Test Suite Variance

In other cases the program overfits the test suite.

This might produce future bugs or coincidental correctness.

This also requires to extend the tests suites.

Test Selection

Select a small set of test cases that most efficiently tests a software system.

The natural intuition is to select diverse test cases

These methods are traditionally based on clustering or similarity-based selection

Automatic Test Generation

Create or extend a test suite automatically, improving its diversity.

This reduces the bias and might activate new behaviours.

It can also reproduce specific bugs.

Diversity and Software Testing

Why does Software Testing need diversity?

What is a proper definition of diversity?

How is diversity applied to Software Testing?

Diversity & Diverse

Diversity: The state or fact of being diverse.

Diverse: of a different kind, form, character, etc.

Information Theory

We need to describe how two objects are different in a general way.

Information Theory shows different ways to measure information content relative of an object and different objects.

Entropy

Entropy measures the uncertainty on a random variable (X).

$$H(X) = \sum_{x \in \mathcal{X}} p(x) \log p(x)$$

Conditional Entropy

We can also measure the uncertainty of a probability distribution given another one. This gives a notion of distance between two random variables (X, Y).

$$H(Y|X) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log p(y|x)$$

Entropy and Uniform

When we want to diversify, we give the same probability to every event.

This probability distribution can only be an $\mathcal{U}$ **distribution**

The **entropy is maximum** iff the random variable follows a $\mathcal{U}$ distribution

Kolmogorov Complexity

Kolmogorov complexity of an object, such as a piece of text (x), is the length of the shortest computer program that produces the object as output.

$$K_{\mathcal{U}}(x) = \min_{p:\mathcal{U}(p)=x} l(p)$$

Conditional Kolmogorov Complexity

Conditional Kolmogorov complexity uses a string as an input (y), and finds the smaller program that transforms it into x.

$$K_{\mathcal{U}}(x|y) = \min_{p:\mathcal{U}(p,y)=x} l(p)$$

Kolmogorov Complexity

Examples of Conditional Kolmogorov Complexity:

$$y = 011101100010011$$
$$x_1 = 010101010101010 \rightarrow \text{"01"} \times 7 + \text{"0"}$$
$$x_2 = 011101100010110 \rightarrow y + \text{"0"}$$

$$K(x_1|y) > K(x_2|y)$$

Normalized Information Distance

It is possible to use the Kolomogorov Complexity to measure the information distance between two strings (x,y):

$$NID(x,y) = \frac{\max\{K(x|y), K(y|x)\}}{\max\{K(x), K(y)\}} = \frac{K(xy) - \min\{K(x), K(y)\}}{\max\{K(x), K(y)\}}$$

NID Attributes

NID is **universal** because it can be shown to be less than any other similarity metric between two strings.

It is **generic** because it does not depend on any particular features of the strings so it can be applied to any type of strings.

It is **non-computable**, but can be approximated.

Normalized Compression Distance

NCD leverages the ability of compressors to approximate Kolmogorov complexity for the approaxiamation of NID.

$$NCD(x, y) = \frac{Z(xy) - \min\{Z(x), Z(y)\}}{\max\{Z(x), Z(y)\}}$$

Normal Compressor

A compressor Z is normal if it satisfies for all strings x, y and z:

$$Z(xx) = Z(x) \text{ and } Z(\epsilon) = 0$$
$$Z(xy) \geqslant Z(x)$$
$$Z(xy) = Z(yx)$$
$$Z(xy) + Z(z) \leqslant Z(xz) + Z(yz)$$

Diversity and Software Testing

Why does Software Testing need diversity?

What is a proper definition of diversity?

How is diversity applied to Software Testing?

Input diversity

Input diversity: classical approach, we uniformly select inputs.

However the inputs have no semantic information.

Input Diversity Approaches

Adaptations of Random Testing improving entropy.

Search-based algorithms using entropy as a fitness.

Universal Hashing on program constraints.

Input Diverse Selection

NCD can be used to calculate diversity on the test suites finding its **diameter**.

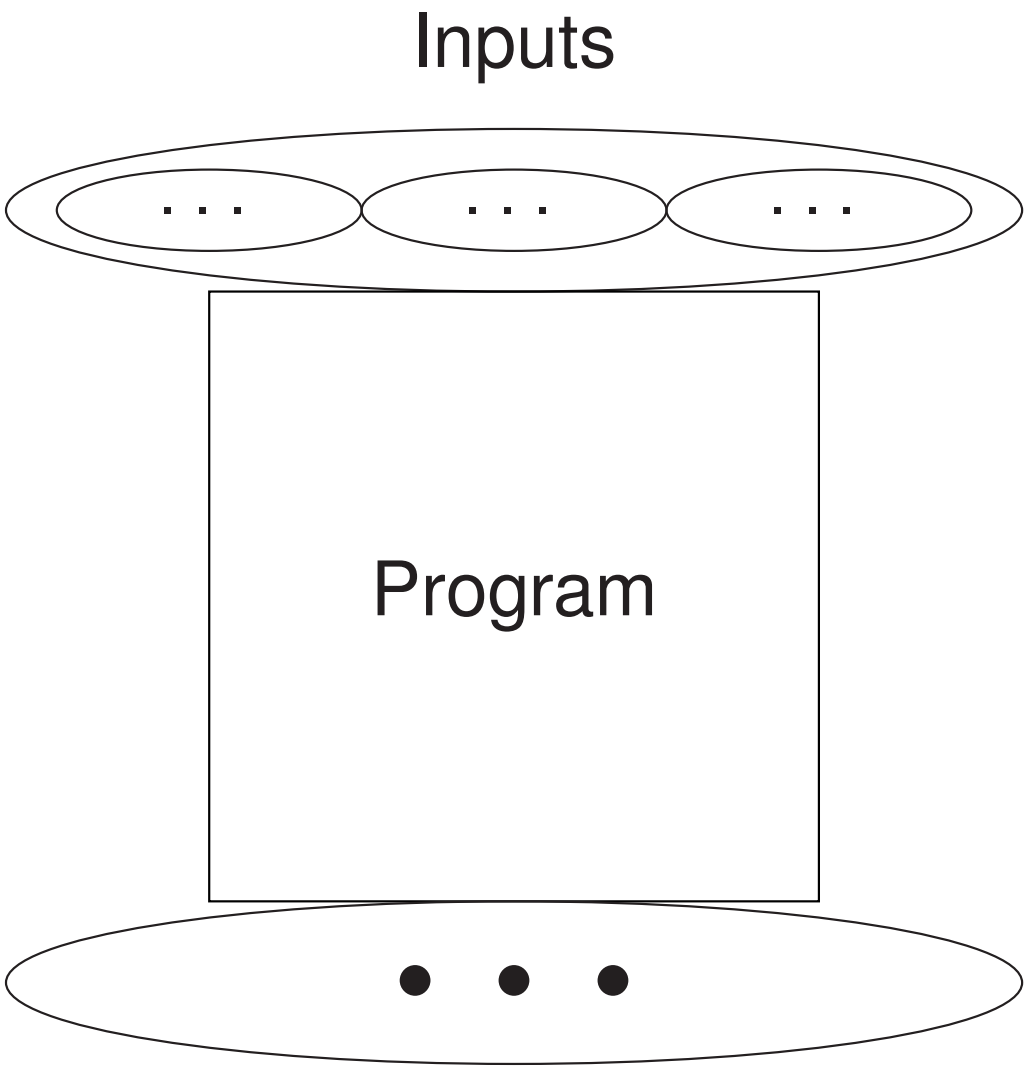
The multiset version of NCD has been used to find small test suites correlated with high coverage.

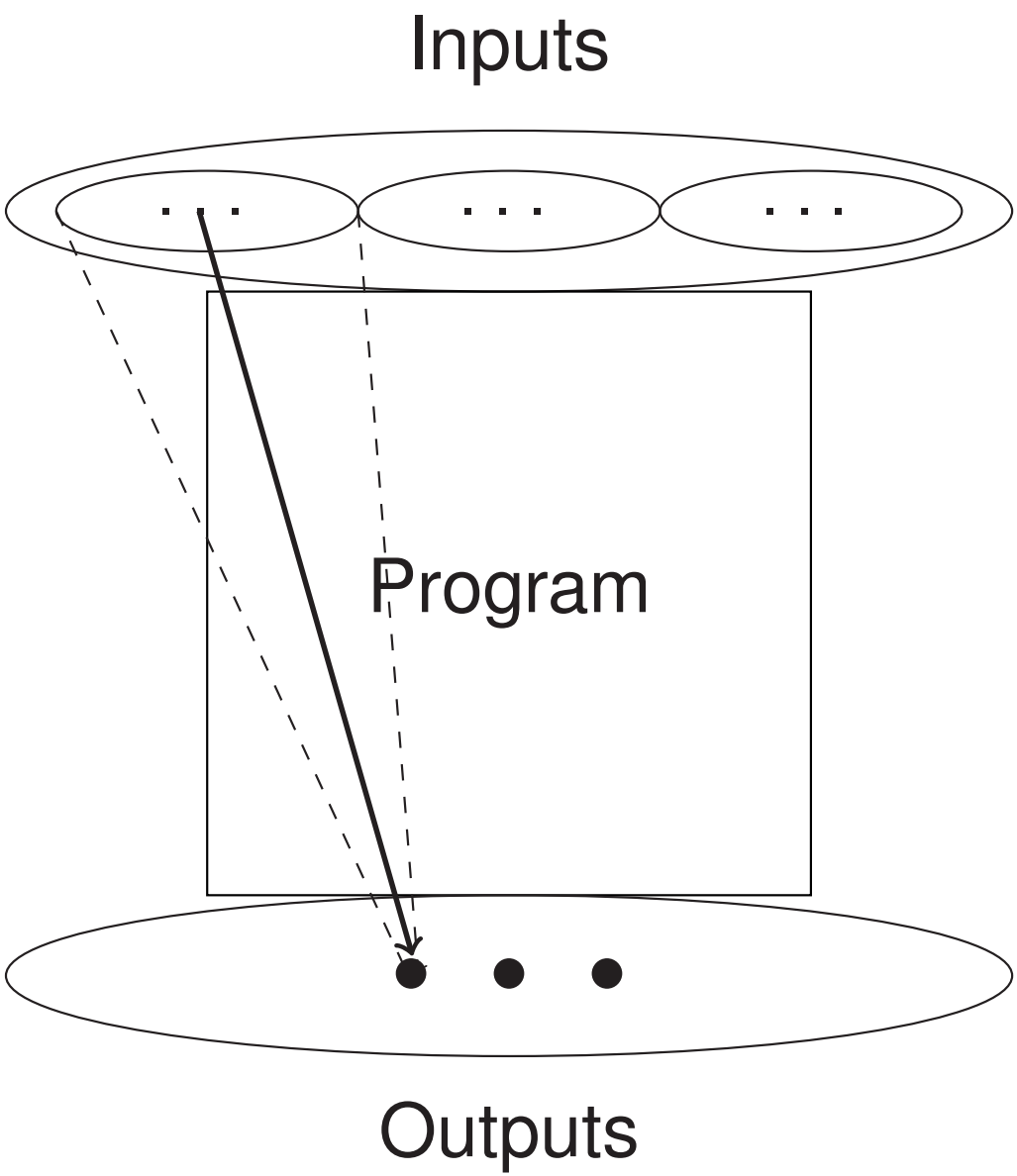
Output Diversity

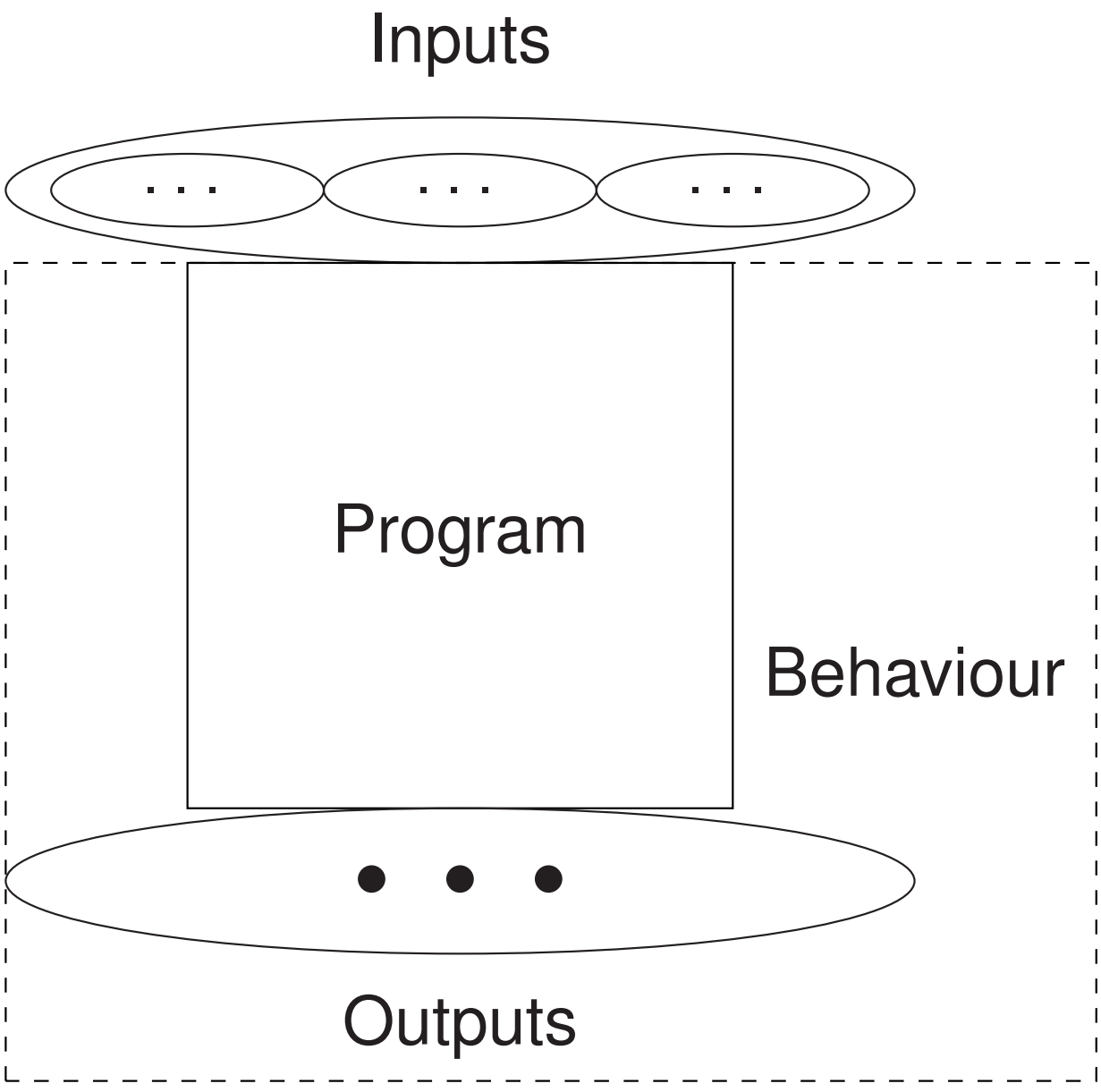
Output diversity aims to create a **diverse test suite** that takes into account **semantic information** of the program.

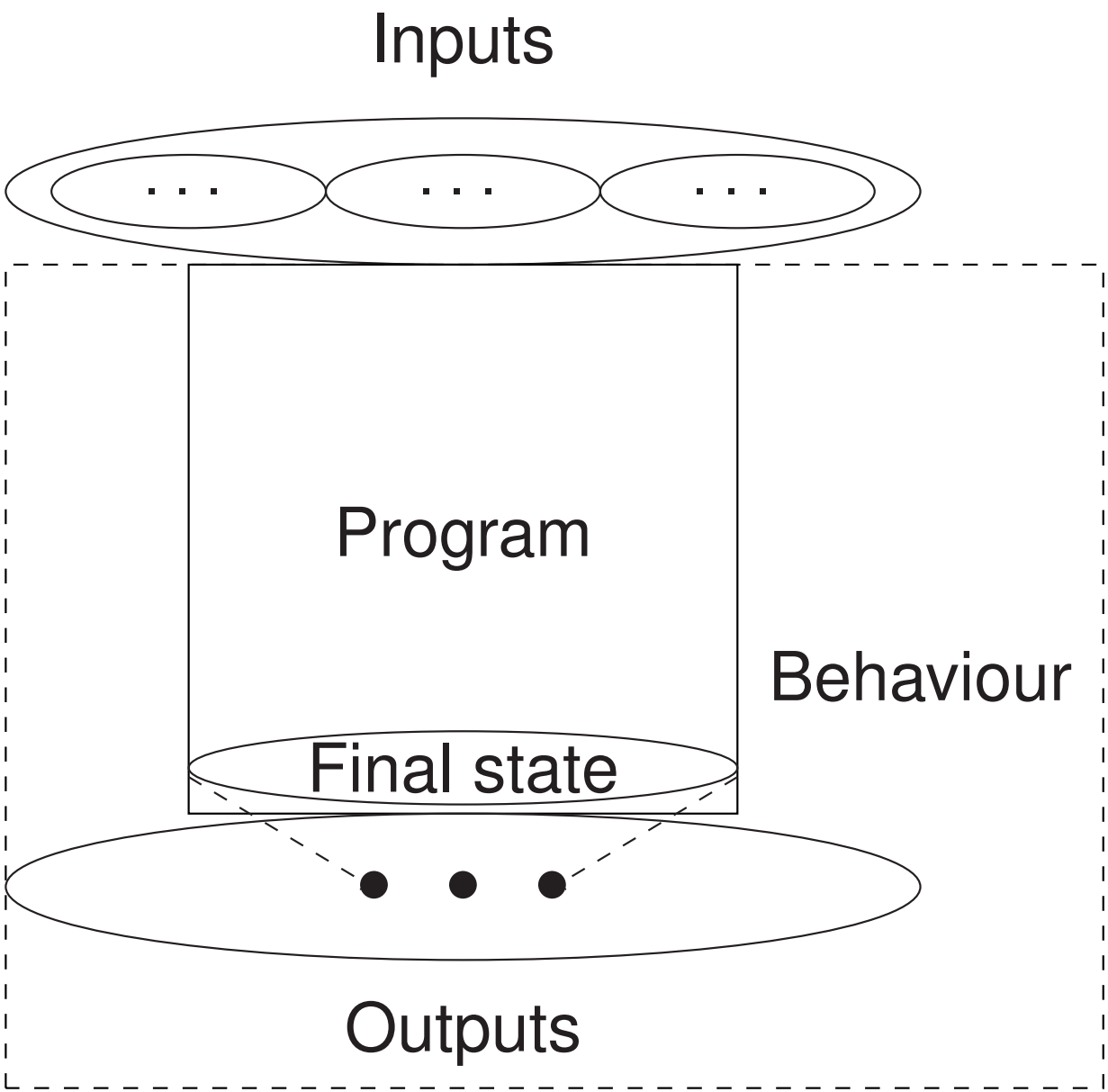
The **squeeziness** directly affects it

The process needs to balance **squeeziness** and **coverage**.


Inputs
Program
Outputs


Inputs
Program
Outputs


Inputs
Program
Behaviour
Outputs


Inputs
Program
Behaviour
Final state
Outputs

Output Diversity Approaches

Output-uniqueness: generate diverse inputs and filter by output uniqueness.

Output-similarity: search for inputs improving a diversity metric based on similarity of outputs.

The Diversity-driven generators

Chakraborty, Meel and Vardi created a diverse input generator based on **SAT solver**

The SUT is considered as a **formula** for a SAT solver (semantics)

They use the solver to create **inputs** through **witnesses** of this formula

The Diversity-driven generators

But the solver uses heuristics and it is **adversarial** in terms of uniformity

They improved uniformity through **universal hash functions**

They divide the inputs space into **cells** and select cells and witnesses uniformly at random

The Diversity-driven generators

We adapted this idea to the **outputs space**, keeping the ability of include extra information

We transform a **program** into a set of **constraints** and, using **bit-vector arithmetic**, we can also adapt their approach to **SMT solvers**

Gödel Testing

Gödel testing aims to create special test inputs following specific structures

The programmer creates a parametrized generator and looks for diverse inputs, combining parameters.

Focused Random Testing

Finds a diverse test suite passing through a specific program point.

Useful for bug reproduction, malware triggering or to study Failed Error Propagation.

It can use search or symbolic execution.

Conclusions

Diversity deals with bias and variance problems of testing.

It can be applied to several different testing strategies.

It can also drive generation methods to create better test suites.

An Overview on Diversity and Software Testing

Héctor D. Menéndez

Department of Computer Science
University College London

3rd April 2018

14TH TAROT SUMMER SCHOOL 2018

on Software Testing, Verification & Validation, UCL, London — 2-6th July 2018

TAROT 2018 Commitees Registration Speakers Accommodation Venue


The Old London


The New London


University College London (UCL)

https://wp.cs.ucl.ac.uk/tarot2018/