Output Diversity

Michele Boreale, David Clark, Daniele Gorla, **Héctor D. Menéndez**

Department of Computer Science
University College London

30th January 2018

Output Diversity

Why do we need output diversity?

How can we generate diverse outputs?

How effective is output diversity?

Output Diversity

Why do we need output diversity?

How can we generate diverse outputs?

How effective is output diversity?

Why Output Diversity?

We want a **diverse test suite** that takes into account **semantic information** of the program.

Why Output Diversity?

We want a **diverse test suite** that takes into account **semantic information** of the program.

What do we understand by diversity?

Why Output Diversity?

We want a **diverse test suite** that takes into account **semantic information** of the program.

What do we understand by diversity?

How the semantic information propagates to the output?

What's diversity

Low Similarity (Normalized Information Distance)

High **Entropy**

Entropy and diverse tests

We want a **generator** creating **diverse tests** for programs

The generator create tests sampling from a **maximum entropy** probability distribution

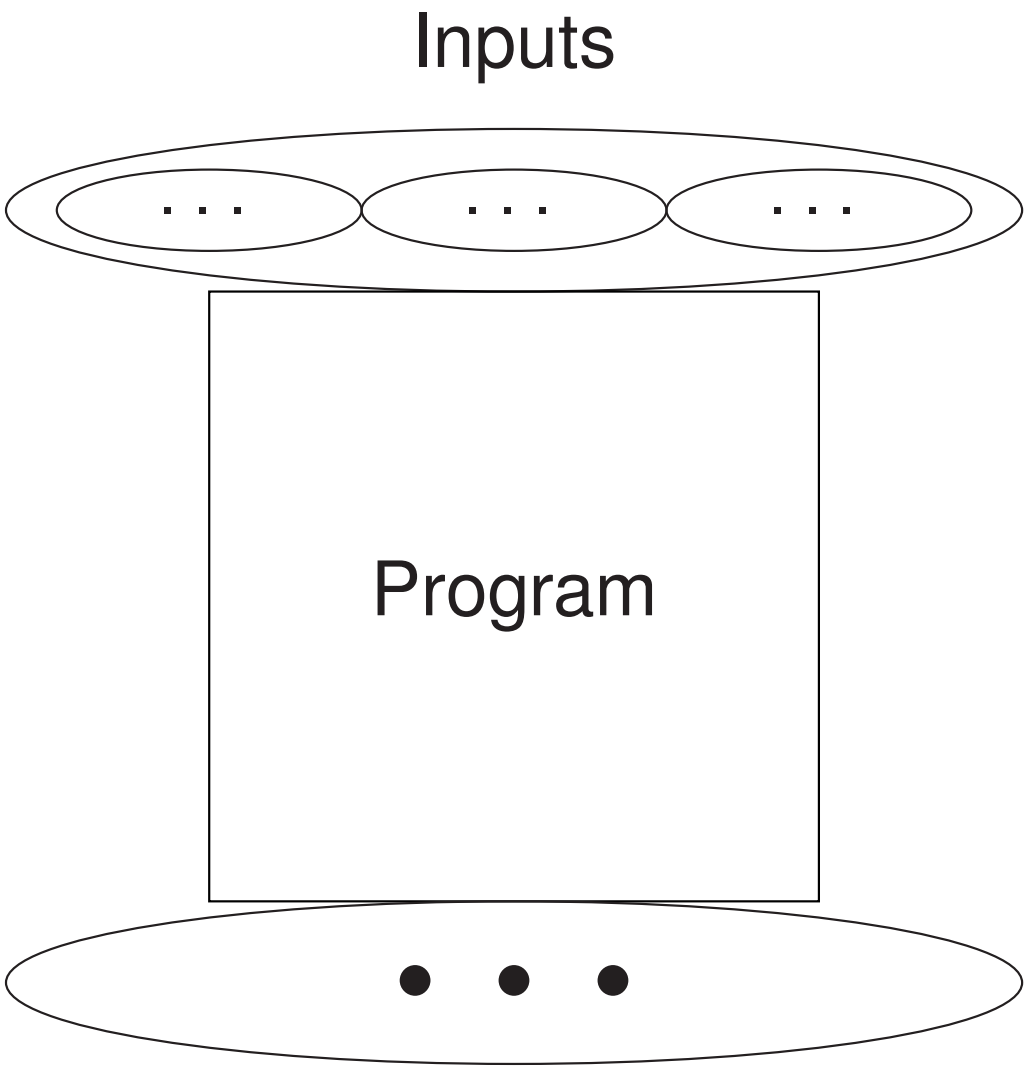
This probability distribution can only be a $\mathcal{U}$ **distribution**

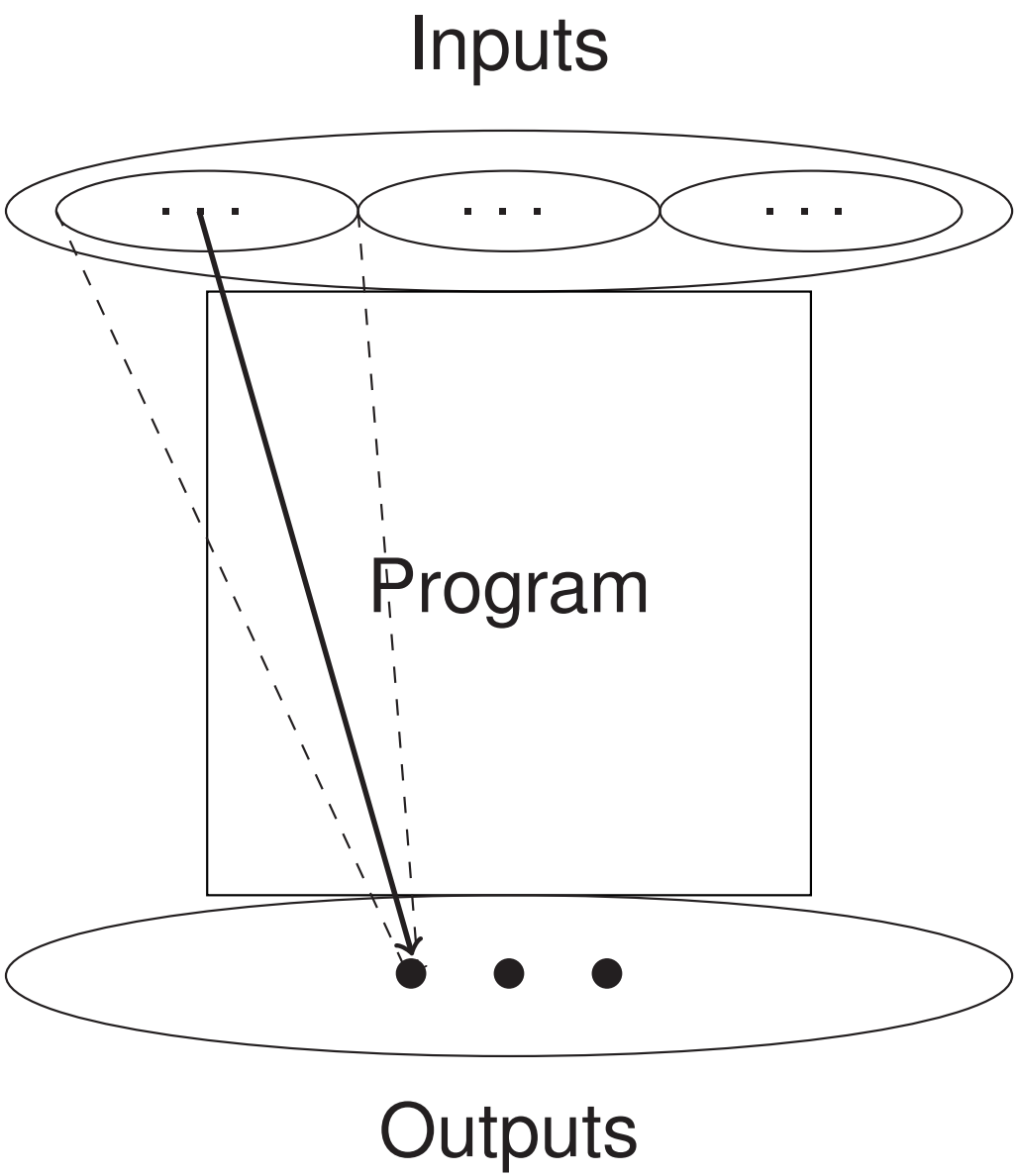
The output semantics

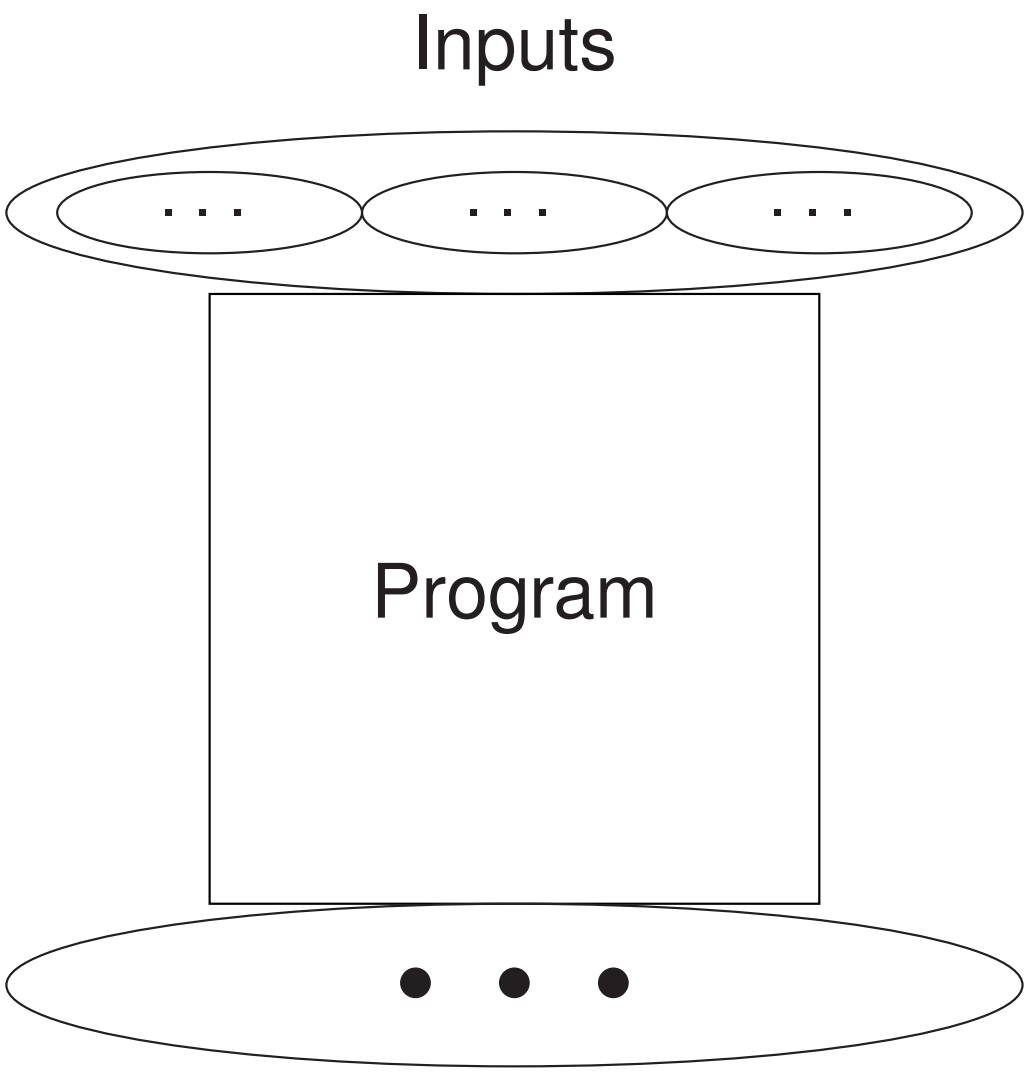
Considering deterministic programs, the I/O behavior works as a map.

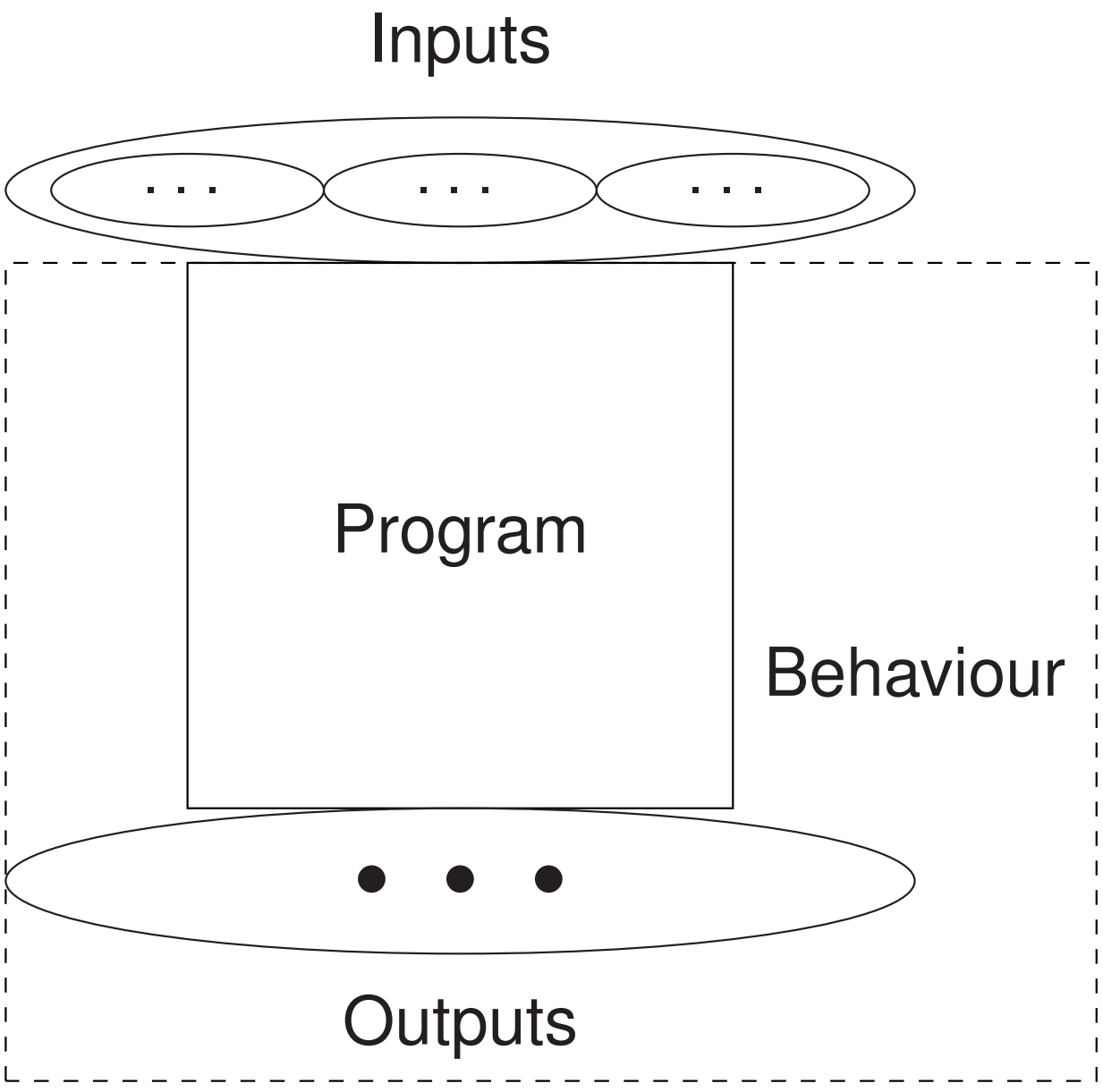
The **squeeziness** directly affects to this map

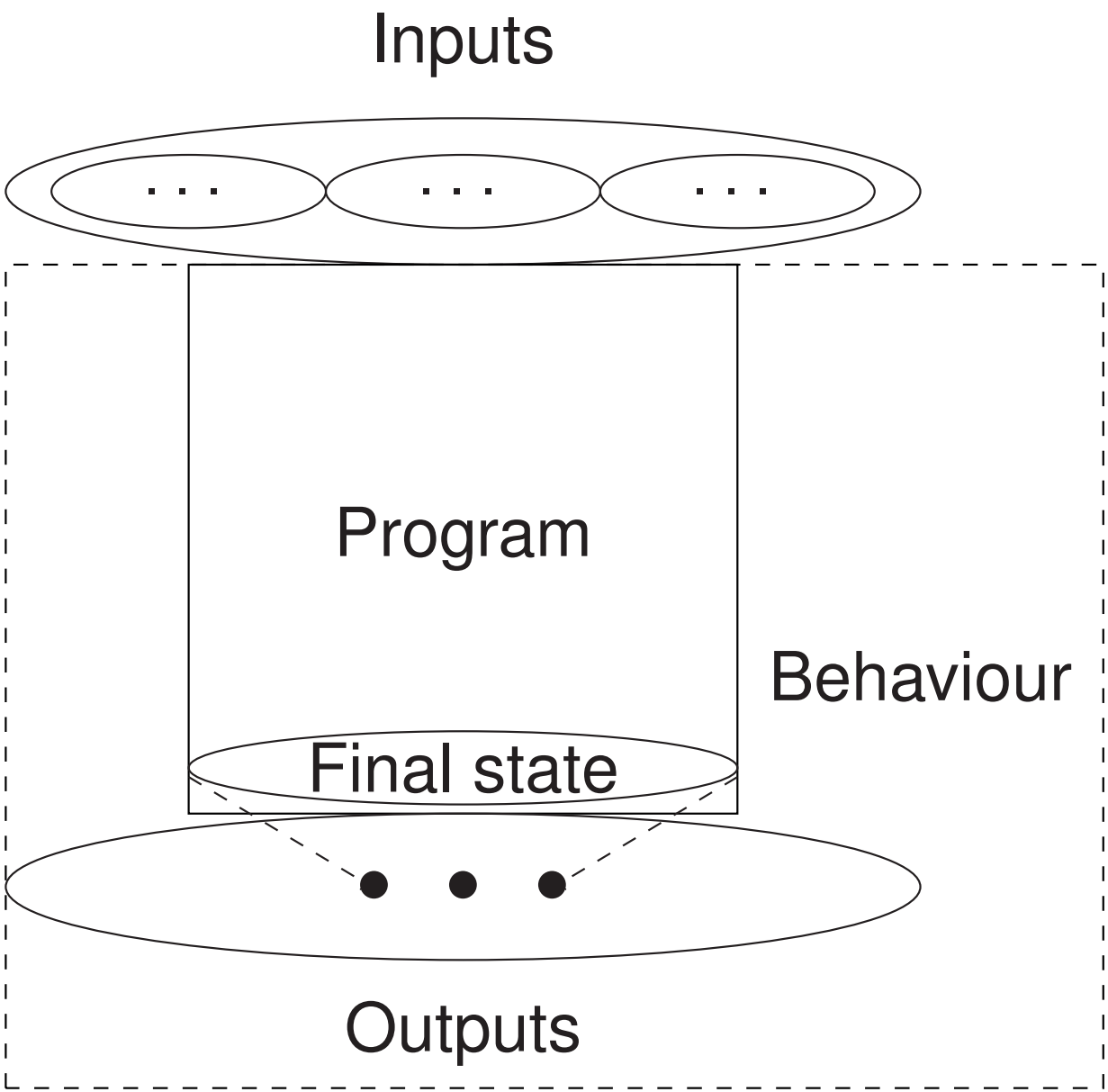
We will need to balance **squeeziness** and **coverage**.


Inputs
Program
Outputs


Inputs
Program
Outputs


Inputs
Program
Outputs


Inputs
Program
Behaviour
Outputs


Inputs
Program
Behaviour
Final state
Outputs

Output Diversity

Why do we need output diversity?

How can we generate diverse outputs?

How effective is output diversity?

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

What happens with the distribution?

Our goal is to maximize entropy, or create an **uniform distribution on the output set**

Roughly speaking, every output has the **same probability** to appear

The effect of the **squeeziness** attacks output uniqueness and search

The output diverse generator (I)

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 output diverse generator (II)

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

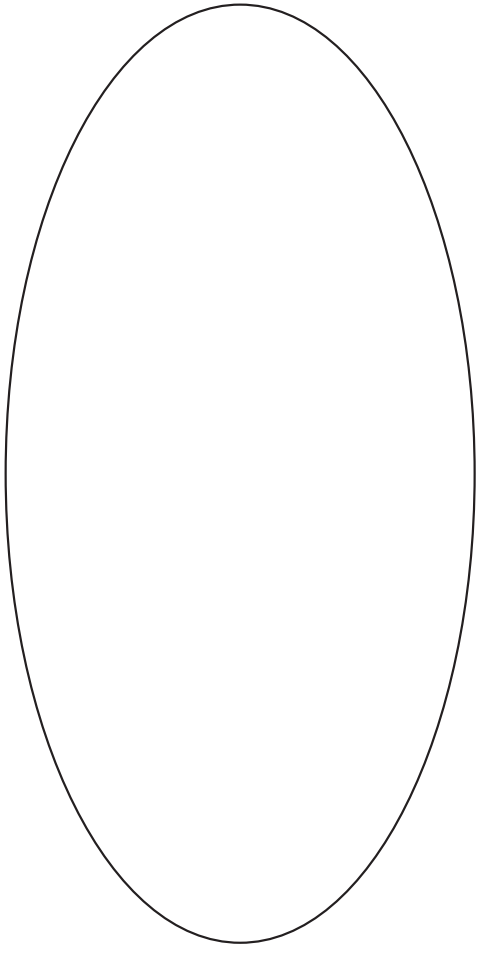
The output diverse generator (III)

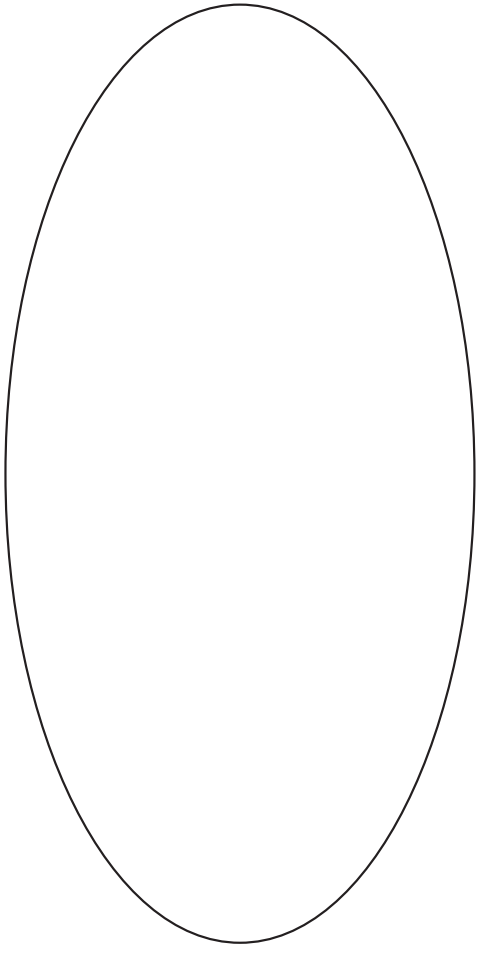
We adapted this idea to the **outputs space**, keeping the ability of include extra information

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

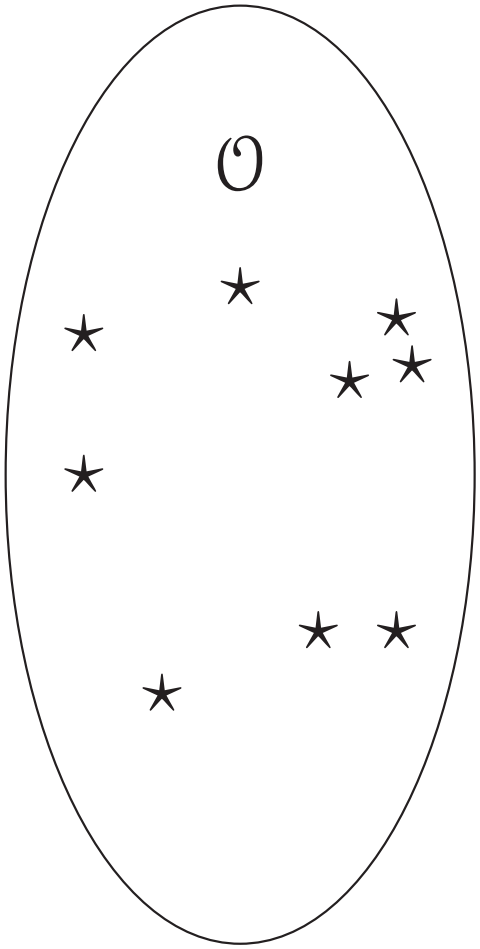
Outs Dom $d(P) \leqslant d(O)$ Proj

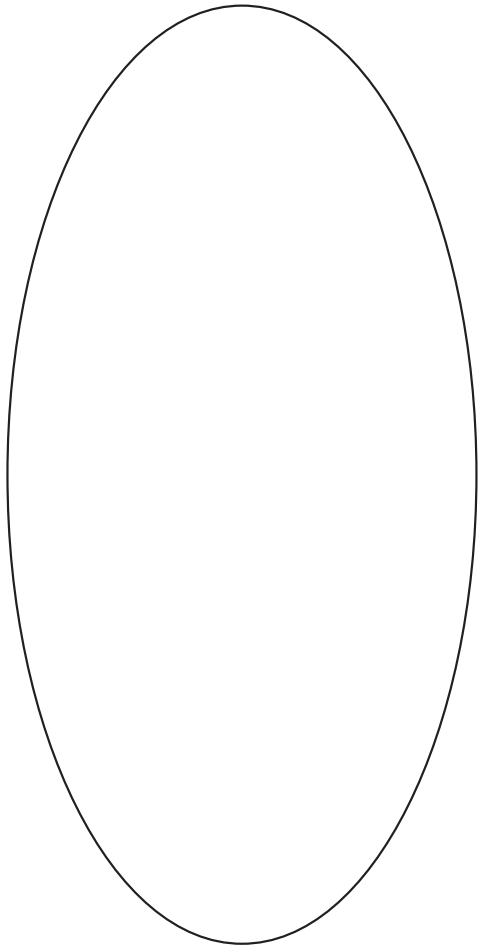
Outs Dom $d(P) \leqslant d(O)$ Proj



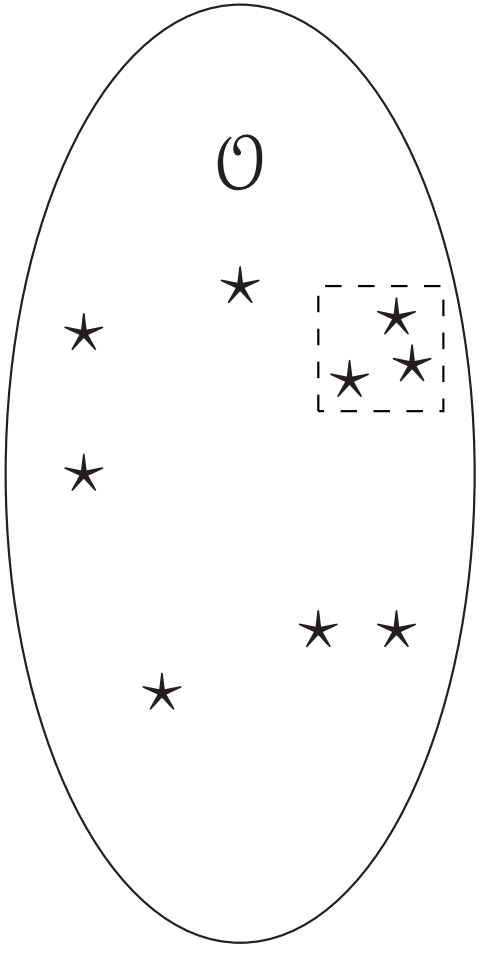


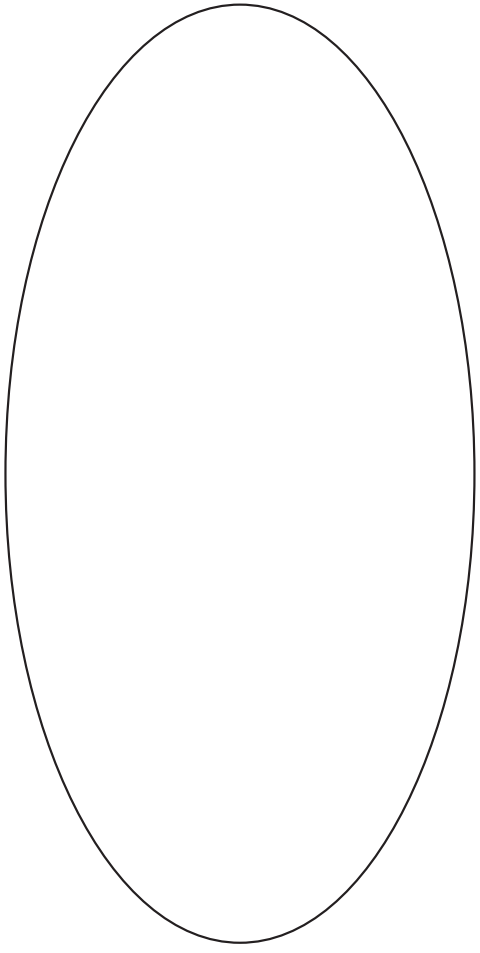
Outs Dom $d(P) \leqslant d(O)$ Proj





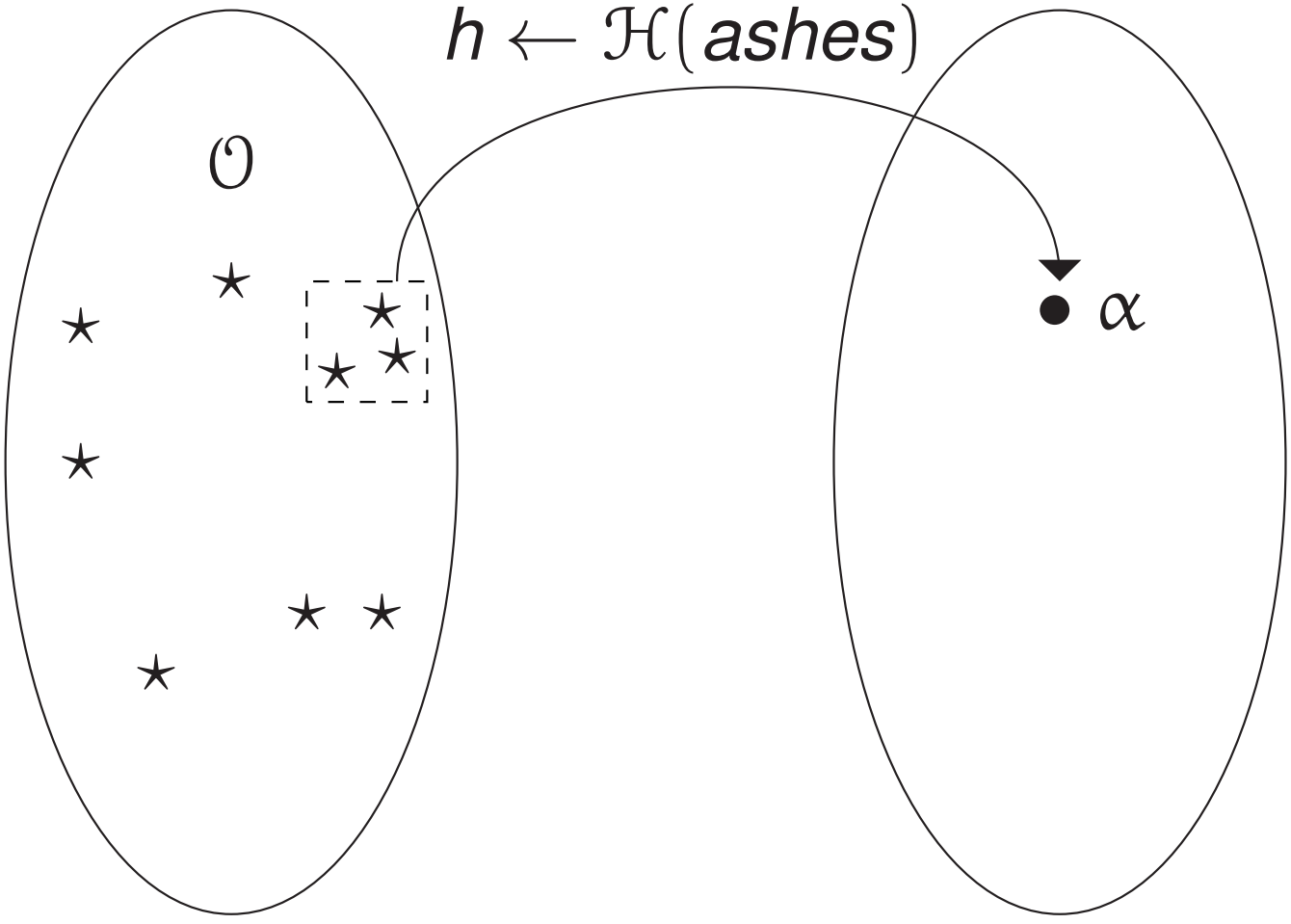
Outs Dom $d(P) \leqslant d(O)$ Proj



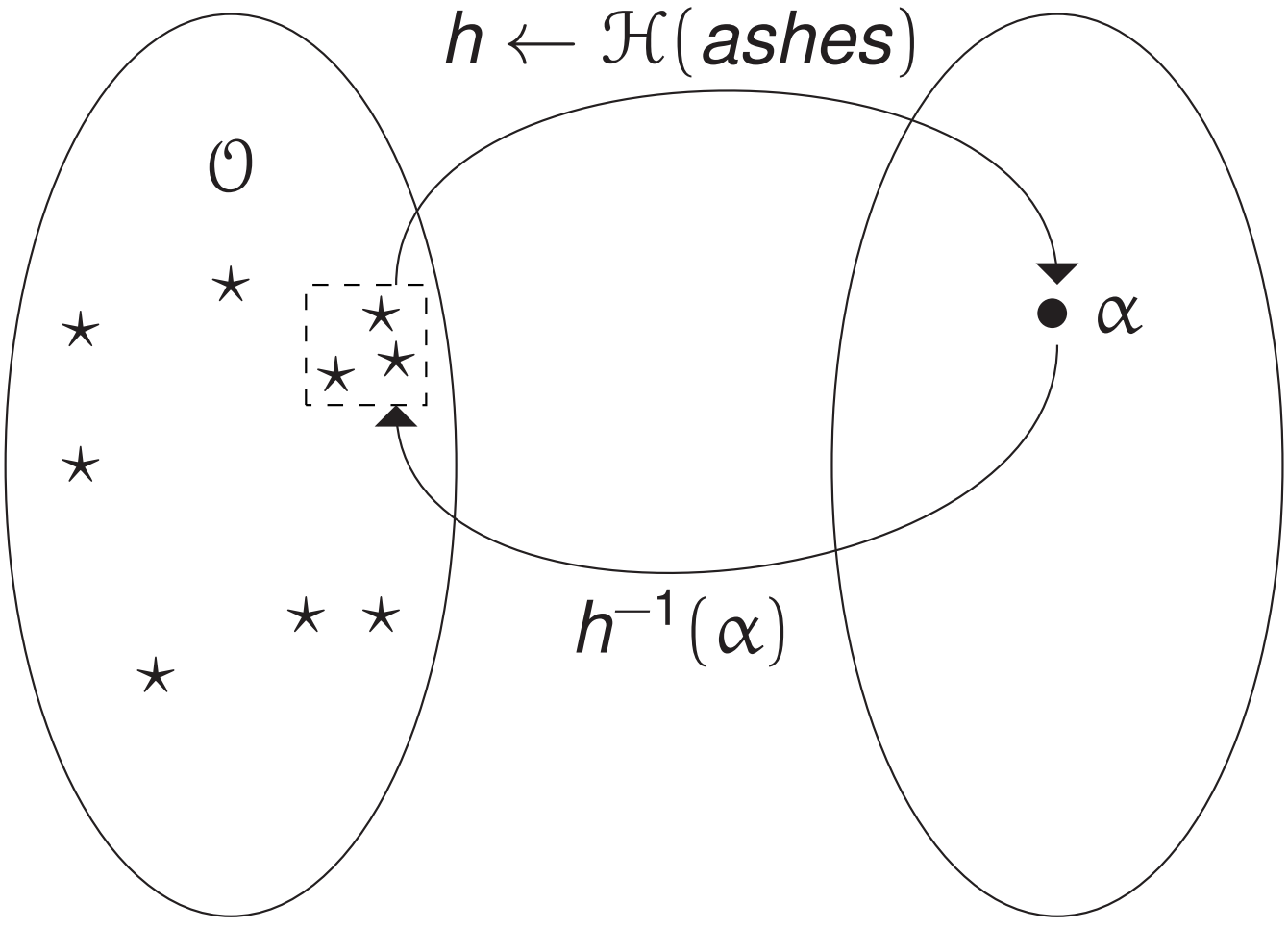


Outs Dom $d(P) \leqslant d(O)$ Proj

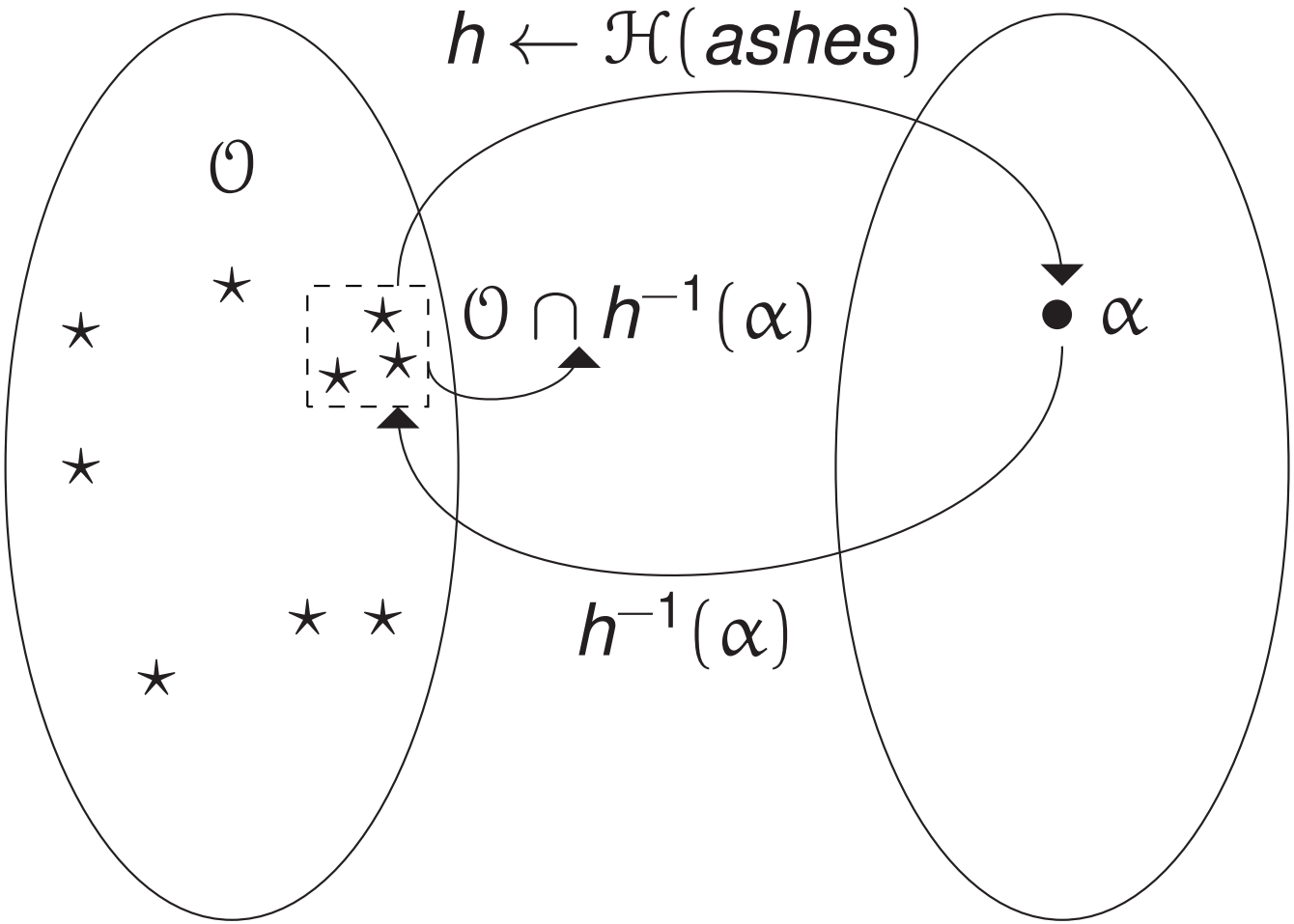
$h \leftarrow \mathcal{H}(ashes)$



Outs Dom $d(P) \leqslant d(O)$ Proj



Outs Dom $d(P) \leqslant d(O)$ Proj



Did we reach uniformity?

No! But, we are closer. We proved **near-uniformity**.

There is a factor on the cell selection process that produces **intersection of cells**

The **Central Limit Theorem** affects within these intersections

Output Diversity

Why do we need output diversity?

How can we generate diverse outputs?

How effective is output diversity?

Experimentation

We used **CodeFlaws** for our experiments

We compared with a **human test suite**, **CBMC** and **CAVM**

Our measure focused on **coverage**, **mutations** and **faults detected**

Coverage

Testing Method	Lines	Branch
Original TS (main)	100% ± 6.9	100% ± 11.6
OD (main)	69.2% ± 19.5	50.0% ± 25.0
OD (main + final state)	100% ± 11.2	100% ± 16.7
CBMC (Lines)	100% ± 12.0	96.9% ± 22.5
CBMC (Branch)	100% ± 12.1	100% ± 23.3
CBMC (Condition)	100% ± 13.9	100% ± 18.3
CBMC (Decision)	100% ± 13.5	100% ± 21.3
CBMC (MCDC)	100% ± 14.4	100% ± 19.1
CAVM	100% ± 0.0	100% ± 12.0

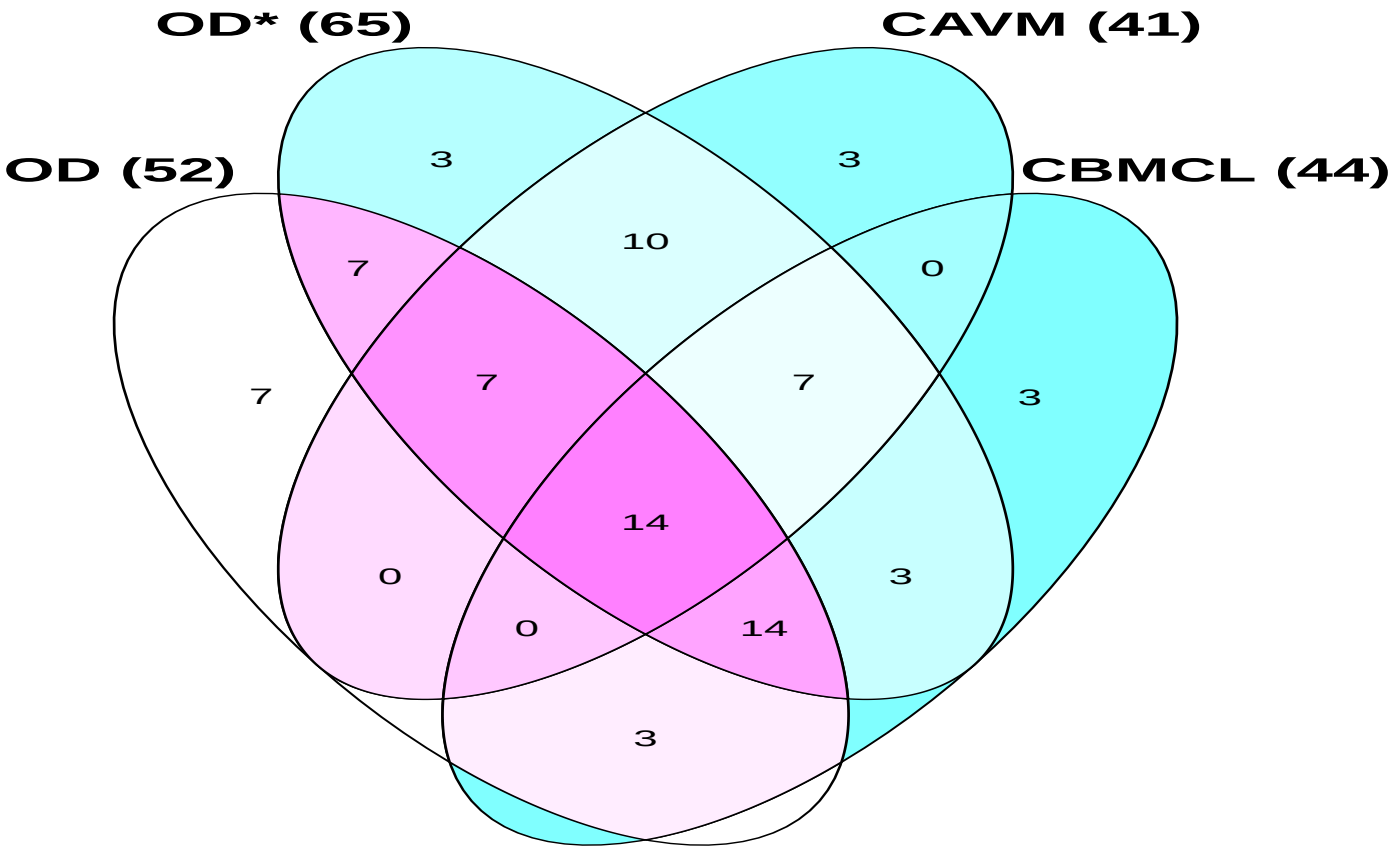
Killing mutants

Testing Method	Killed	Not Killed
Original TS	90.80%	9.20%
OD	71.43%	28.57%
OD (+ final state)	87.50%	12.50%
CBMC (Line)	78.57%	21.43%
CBMC (Branch)	78.57%	21.43%
CBMC (Condition)	85.71%	14.29%
CBMC (Decision)	81.48%	18.52%
CBMC (MCDC)	85.71%	14.29%
CAVM	77.78%	22.22%

Fault detection

Testing Method	Found	Not Found
Original TS	98%	2%
OD	52%	48%
OD (+ final state)	65%	35%
CBMC (Line)	44%	56%
CBMC (Branch)	31%	69%
CBMC (Condition)	37%	63%
CBMC (Decision)	31%	69%
CBMC (MCDC)	34%	66%
CAVM	41%	59%

Fault detection



Output Diversity

Why do we need output diversity?

How can we generate diverse outputs?

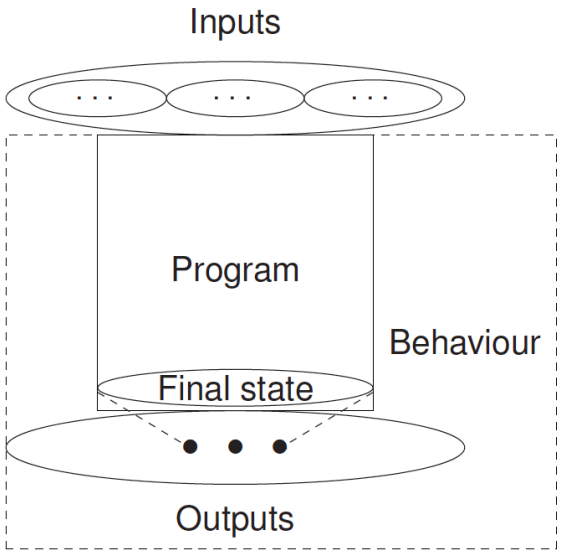
How effective is Output Diversity?

Output Diversity

Why do we need output diversity?

How can we generate diverse outputs?

How effective is Output Diversity?

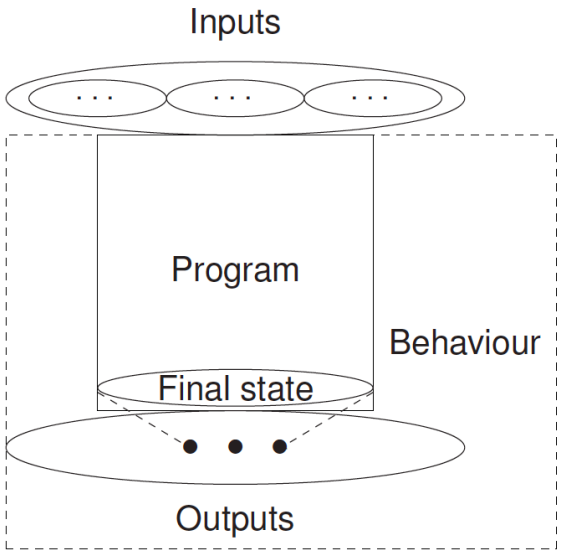


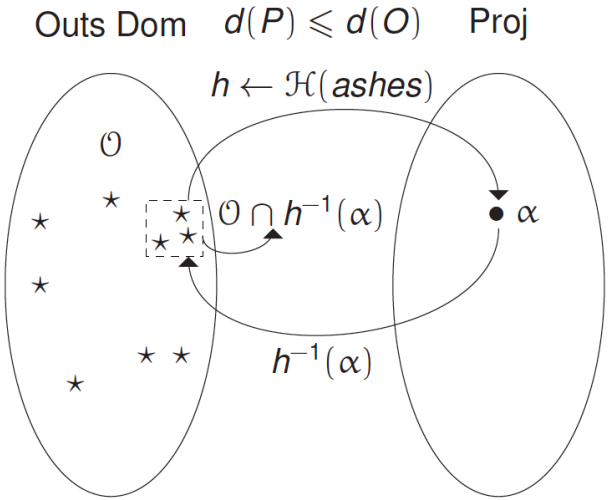
Output Diversity

Why do we need output diversity?

How can we generate diverse outputs?

How effective is Output Diversity?



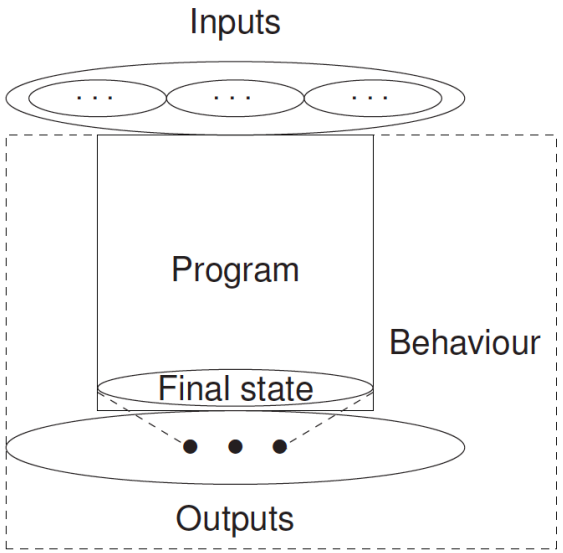


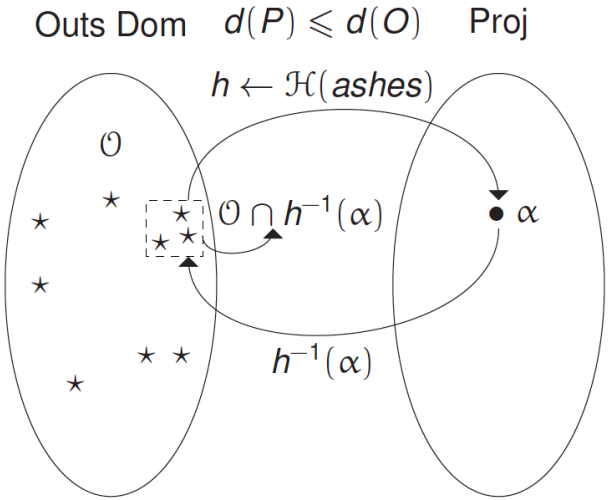
Output Diversity

Why do we need output diversity?

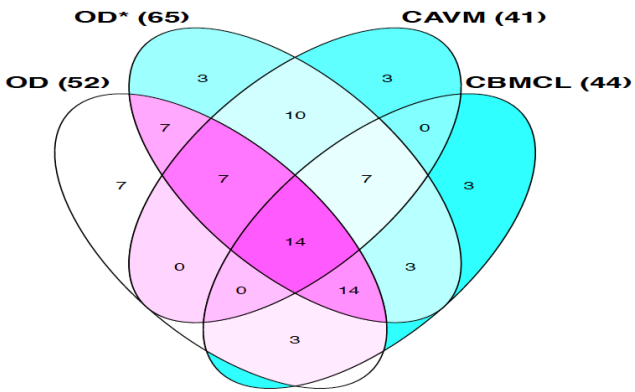
How can we generate diverse outputs?

How effective is Output Diversity?





Fault detection



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/