
DTU

Program Analysis – Group 21

Members:

- Andrea Onorato
- Enrico Tuda
- Paul Nelson Becker
- Thor-Haakon Jørgensen

Introduction

Is concolic testing more effective than random testing for verifying a function's output range?

Bonus Research Question: Can we improve the concolic analysis to verify a function's output range by skipping loops in the analysis?

Concolic Testing

- Input range constraints
- Use concolic analysis to explore execution paths
- Find symbolic return
- Evaluate symbolic return with range and path constraints
- Use z3 to create path constraints and generate inputs

Inputs	Path Constraints	Symbolic Return	Return Check
$q = 100,$ $cE = 3$ $cG = 2$ $e = 1$	$cG \geq 1 \wedge cG \leq 10$ $\wedge e \geq 1 \wedge e \leq 10$ $\wedge q > 0 \wedge cE > 0$ $\wedge e \leq cG$ $\wedge \neg(qC * cE \leq 120)$	$qC * cE - 2e$	$\neg(qC * cE - 2e >= 0)$

Random Testing Tool

Pseudo-code:

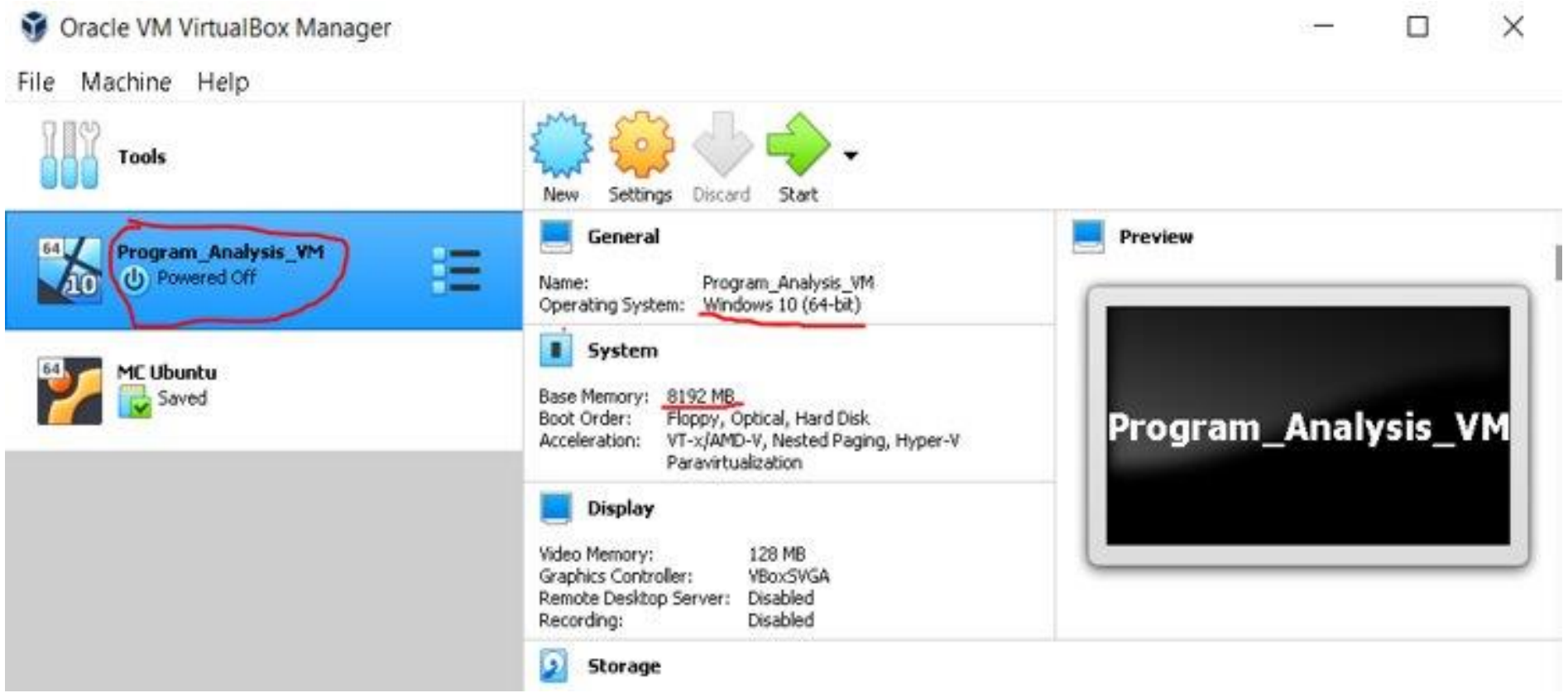
- StartTime()
- while (output in range):
- output = analyzedMethod.invoke(random_inputs())
- StopTime()

No memory tool

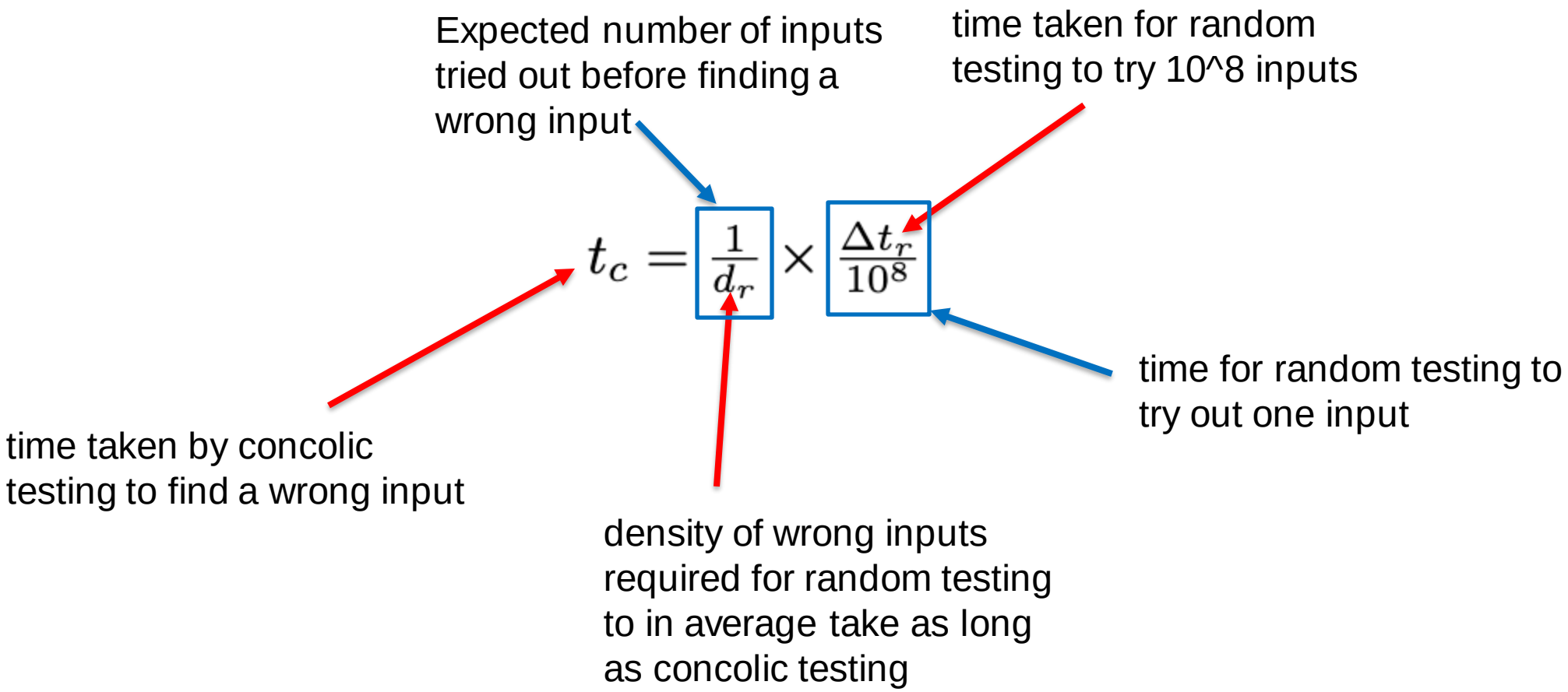
Quality of random inputs

Low sample variance

Testing environment



Evaluation – out of range



Evaluation – example functions

Calculate Efficiency

- 4 inputs
- No loop
- Ca. 100 out of range input combinations
- Output range: >=0

No loop

- 3 inputs
- No loop
- Few out of range input combinations
- Output range: >0

Short loop

- 2 inputs
- 10 iterations loop
- 6 out of range input combinations
- Output range: !=0

Long loop

- 2 inputs
- 1000 iterations loop
- 6 out of range input combinations
- Output range: !=0

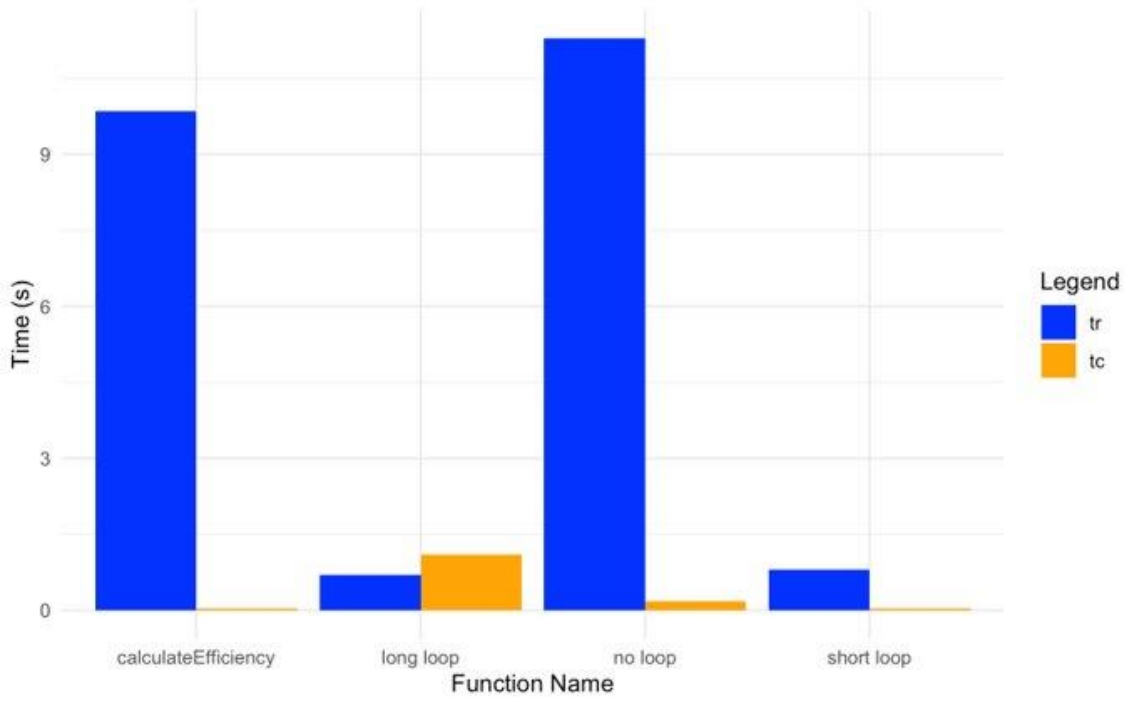
Evaluation – out of range

- Long loop is slow with concolic testing
- calculateEfficiency and no loop take longer per input using random testing
- Short loop and long loop take similar time per input using random testing

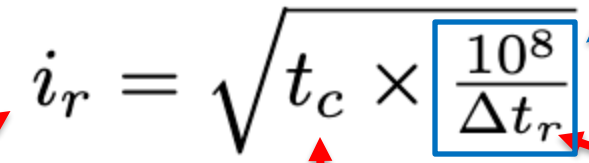
Function Name	Δt_r	tc	d_r
no loop	5.977	0.183	$\frac{327}{10^9}$
calculateEfficiency	7.698	0.034	$\frac{2264}{10^9}$
short loop	4.187	0.035	$\frac{1196}{10^9}$
long loop	4.084	1.098	$\frac{37}{10^9}$

Evaluation – out of range

- range 0 to 10000 for each input
- dependency on number of inputs a function takes



Evaluation – no out of range



inputs random testing can try out per second

input range for each of the two inputs that random testing can try out in the time taken by concolic testing

time taken by concolic testing to return no out of range outputs possible

time taken for random testing to try 10^8 inputs

Evaluation – no out of range

Function Name	Δt_r	tc	i_r
short loop (fixed)	4.099	0.518	3554
long loop (fixed)	4.136	134.443	57013

Discussion of the technique and alternatives

Static analysis/abstract interpretation

Sign abstraction:only positive negative and zero in the analysis

Range abstraction: values of the variables in certain ranges, could lead to incorrect outputs because of over approximation

Pro: Complete

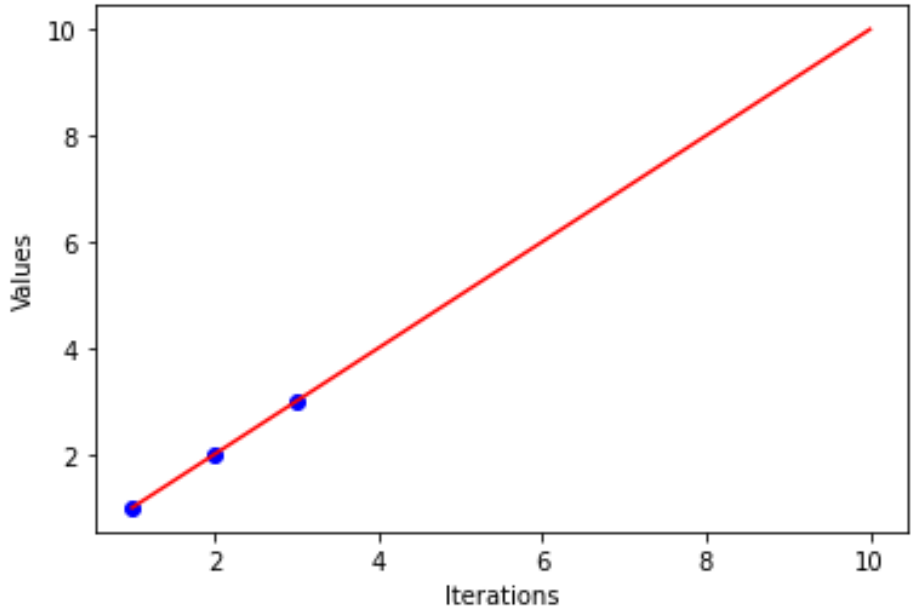
Cons: not feasable for complex programs, not sound

Bonus Research Question: Skip Loops

- The problem: Loops are largely increasing time expenditure
- Our plan: Skip loops in analysis
- This is impossible
- New Plan: Predict loop behaviour to skip loops

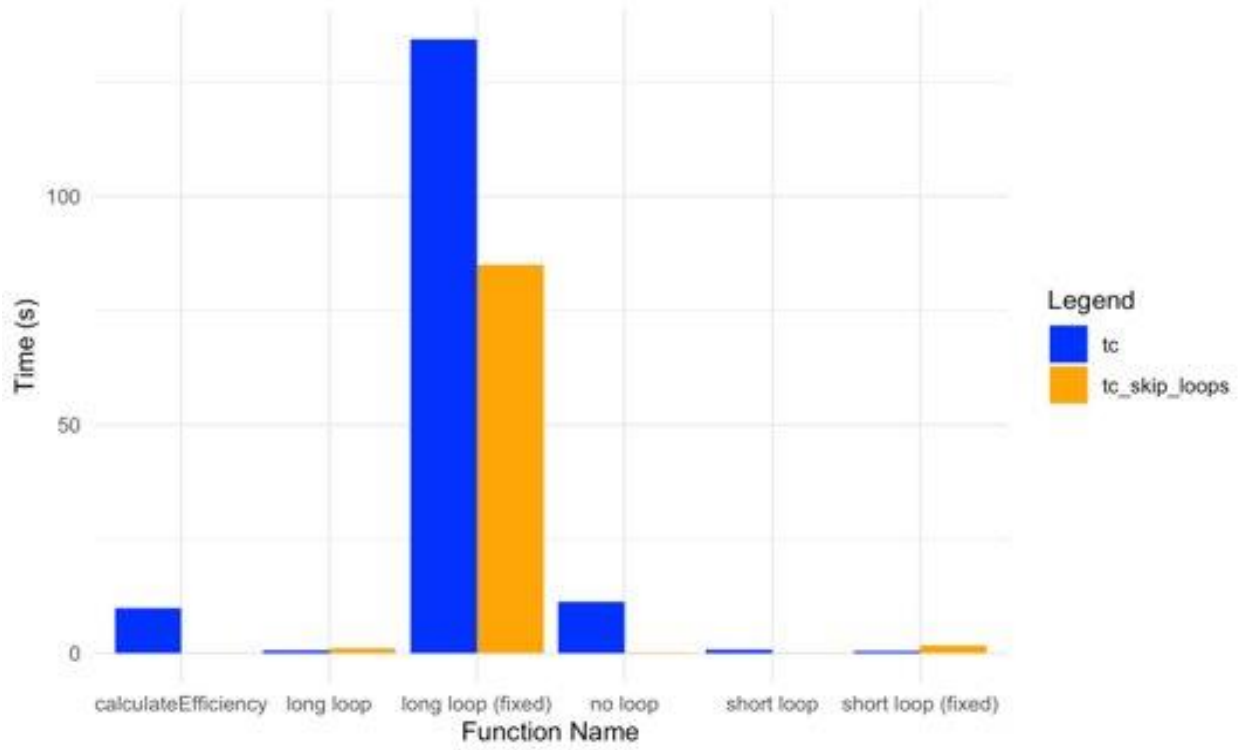
Skip Loops Approach and Implementation

- Use linear regression to skip loops
- Keep track of states at if statements
- Use states to predict iterations and outputs
- Skip loop and update path constraints



Skip Loops Evaluation

- Faster for longer loops vs random testing
- Only works for very simple functions
- Not sound and not complete
- Overall, not a good idea



Conclusion

In General → Random testing: sound ,no complete

In General → Concolic testing: complete, sound, depends on the program

When is random testing better? → Too many wrong inputs

When is random testing better? → Complex programs, too many different paths

When is concolic testing better? → Low code complexity, not too many different paths, few wrong inputs

Can we skip loops? NO → No accuracy, no soundness and completeness

Questions?