General Predicate Testing

Author:
Andi Péter, MsC

Supervisor:
Kovács Attila, PhD

Agenda

- Motivation
- The General Predicate Testing Method
- My GPT implementation
  - GPT Lang
  - Disjunctions
  - Test case reduction
  - Graph reduction algorithms
  - Results of measurements
- Summary
- Q&A

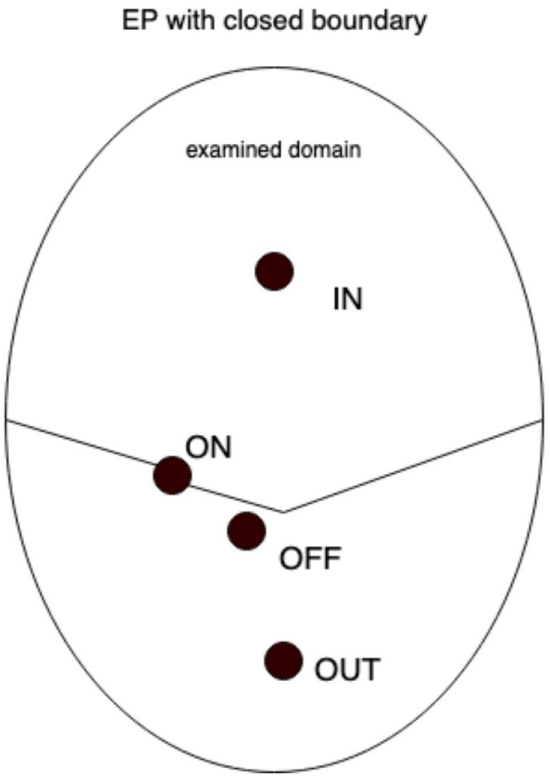
What are predicate errors?

- We have a spaceship going towards the Moon
- If the spaceship would point towards the Earth,
  and we're not above 1000 meters, activate self destruct sequence

```
if(0 <= angle && altitude < 1000) {
  self_destruct();
}
```

Boundary Value Analysis

- Black-box testing method
- Partitioning predicates to equivalence classes
  - All values inside a partition has the same behaviour
  - 0 < x => -830, 0, 1, 567
- Mostly manual method
  - Some tools exist, but not widespread



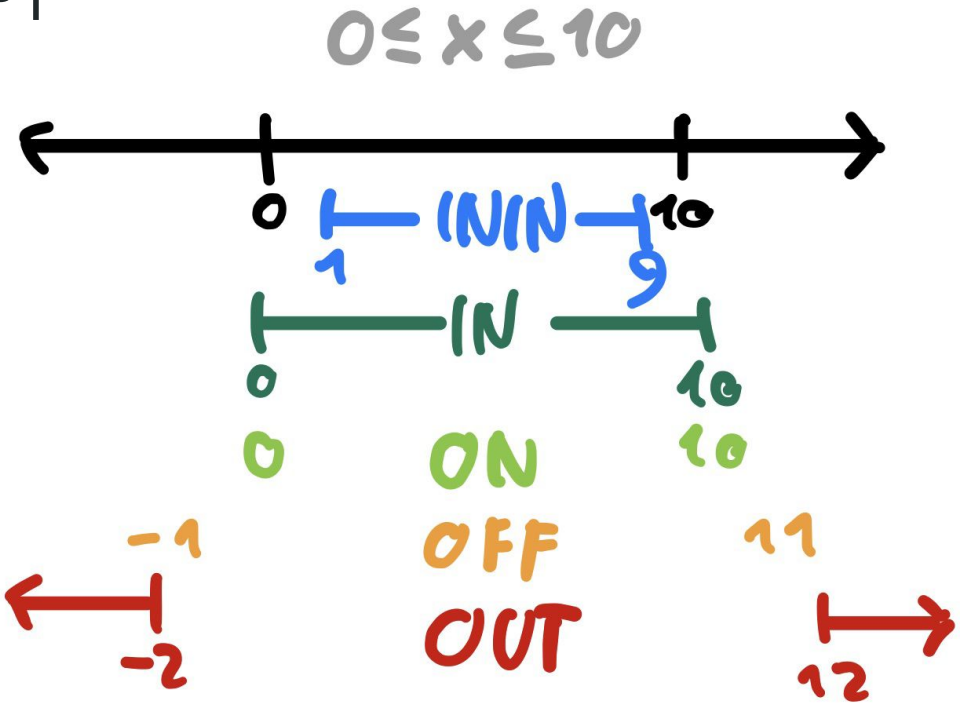
The General Predicate Testing Method

The General Predicate Testing Method

- Proposed by Kovács Attila and Forgács István
  - Paradigm Shift in Software Testing: Practical Guide for Developers and Testers
- Systematic extension of BVA
- Uses intervals instead of single points
  - Overlapping test cases can be merged
  - Reduces the total number of test cases

Equivalence Partitioning in GPT

- ININ: One step inside IN
- IN: Acceptable values
- ON: First acceptable values
- OFF: First not acceptable values
- Out: Not acceptable values (except OFF)



Precision: smallest step between values

Test Case Generation

NTuple

$x < 10$ && y in [0, 20] && z == true => { x: (-Inf, 10) y: [0, 20] z: true }

- Acceptable cases (ON, IN, ININ)

```
IN:   { x: (-Inf, 9.99], y: [0, 20], z: true }
ININ: { x: (-Inf, 9.98], y: [0.01, 19.99], z: true }
ON:   { x: 9.99, y: 0 and 20, z: true }
```

- Not Acceptable cases (OFF, OUT)

```
OFF:
   x:   { x: 10, y: [0, 20], z: true }
   y1:  { x: (-Inf, 9.99], y: -0.01, z: true }
   y2:  { x: (-Inf, 9.99], y: 20.01, z: true }
   z:   { x: (-Inf, 9.99], y: [0, 20], z: false }

OUT:
   x:   { x: [10.01, Inf), y: [0, 20], z: true }
   y1:  { x: (-Inf, 9.99], y: (-Inf, -0.02], z: true }
   y2:  { x: (-Inf, 9.99], y: [20.02, Inf), z: true }
   z:   { x: (-Inf, 9.99], y: [0, 20], z: false }
```

Test case reduction

- T1: { x: [0, 10] }
- T2: { x: (-10, 5) }
- T3: { x: [0, 200] }

- The x = 2 test point would cover all three test cases
- We can merge overlapping test cases by intersecting their intervals
- The more test cases we have the harder it is to do by hand

Research areas - The goals of my thesis

- Creating an algorithm for the GPT Method
- Creating an algorithm for test case reduction
- Designing a specification language for GPT
- Handling negation, multiple predicates for a variable
  - Using multiintervals
- GPT can only work with conjunctions
  - How could we handle disjunctions?

My GPT Implementation

General predicate test description

```
var age: int
var service: int

// R1
if(age < 18 && service < 30)
if(age >= 60 && service < 30)
if(service >= 30 && age < 60 && age >= 18)

// R1-2
if(service >= 30 && age >= 60)

// R1-3
if(service >= 15 && age < 45 && age >= 18 &&
service < 30)
if(age >= 45 && service < 30 && age < 60)
```

Notebook

```
Paid vacation days
R1 The number of paid vacation days depends on
age and years of service. Every employee
receives at least 22 days per year.
Additional days are provided according to the
following criteria: R2-1 Employees younger than
18 or at least 60 years, or employees with at
least 30 years of service will receive 5 extra
days.
R2-2 Employees of age 60 or more with at least
```

Generate Tests

Open user guide

Generated test cases

☐ Show interval values

∗ can be any value you like

	age	service
1	17.0	29.0
2	45.0	31.0
3	16.0	15.0
4	16.0	30.0
5	17.0	31.0
6	18.0	15.0
7	59.0	30.0
8	61.0	28.0
9	60.0	29.0
10	46.0	15.0
11	59.0	29.0
12	18.0	14.0
13	60.0	30.0
14	61.0	31.0
15	19.0	31.0
16	18.0	30.0
17	19.0	16.0
18	44.0	15.0
19	18.0	29.0
20	18.0	13.0
21	45.0	29.0
22	44.0	29.0

GPT Lang

```
var vip: bool
var price: num(0.01)
var second_hand_price: int

if(vip == true && price < 50) {
   if(second_hand_price == 2)
   if(second_hand_price in [6,8])
}

if(vip == false && price >= 50)
else if(vip == true || !(price >= 50 && second_hand_price >= 20)) {
   if(30 < price && 60 < second_hand_price)
}
else
```

Parsing GPT Lang

```
if(x == 1) {
   if(y == 2)
   if(y == 3)
} else if(x < 10)
else {
   if(z == 0)
}
```

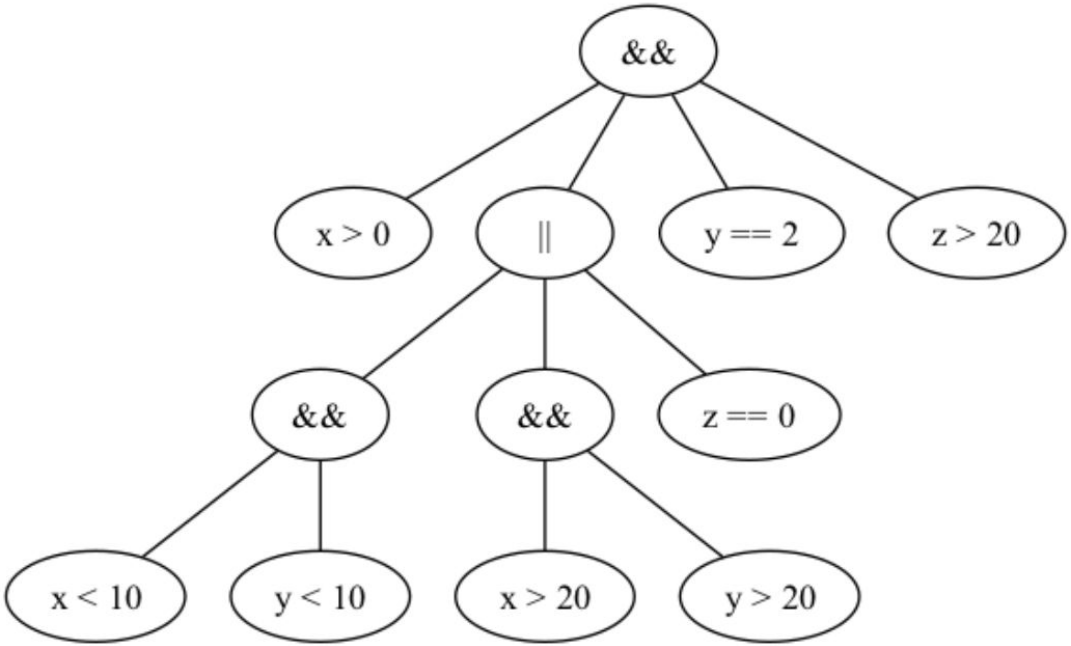
```
x == 1
x == 1 && y == 2
x == 1 && y == 3
!(x == 1) && x < 10
!(x == 1) && !(x < 10)
!(x == 1) && !(x < 10) && z == 0
```

GPT Lang

Intermediary Representation

Flattened Intermediary Representation

x > 0 && (x < 10 && y < 10 || x > 20 && y > 10 || z == 0) && y== 2 && z > 20



Multiintervals

- Multiple intervals, which behave as an interval
  - (-Inf, 0] [5,6] (10, 50) [52, Inf)
- Negating an interval creates a multiinterval
  - x not in [0, 10] => (-Inf, 0) (10, Inf)
- Multiple predicates on a variable create a multiinterval
  - 0 < x && x <= 100 && x not in [18, 60] => (0, 18) (60, 100]

Disjunction

- x || y
  - x
  - !x && y
- We are black-box testing.

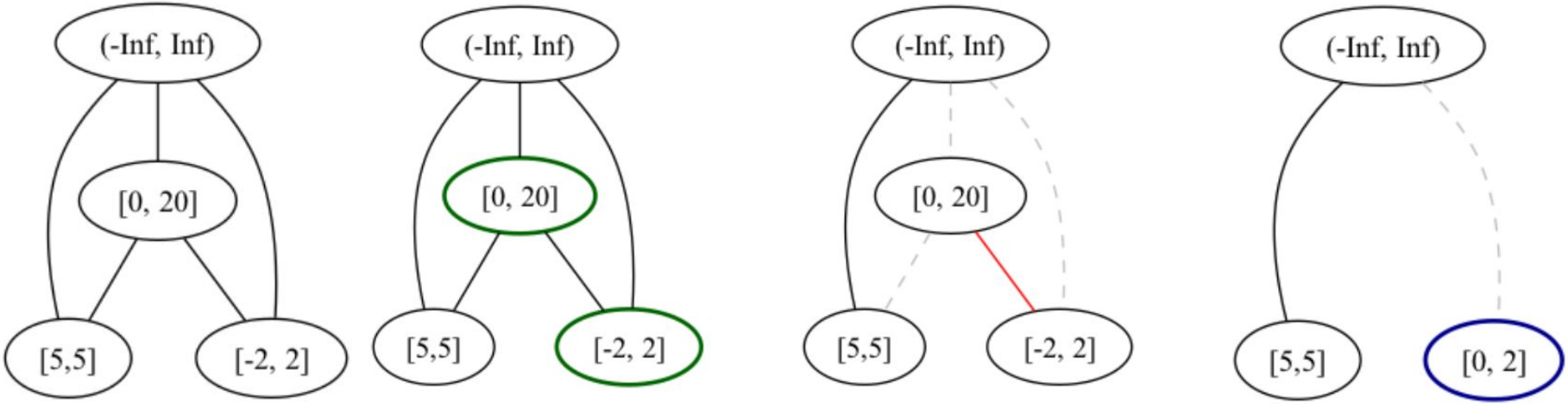
  What if it's implemented as y || x?
  - y
  - !y && x

Example: `(a || b) && (c || d)`

```
a && c
a && !c && d
a && d
a && !d && c
!a && b && c
!a && b && !c && d
!a && b && d
!a && b && !d && c
b && c
b && !c && d
b && d
b && !d && c
!b && a && c
!b && a && !c && d
!b && a && d
!b && a && !d && c
```

Test case reduction - Graph reduction

- If test cases intersect, we can combine them
- Let’s imagine it as a graph:
  - Test cases will be the nodes
  - Two nodes will have an edge if they intersect



Graph Reduction

- There can be non-optimal joins
- Irreversible change to the graph
- NP-hard problem (according to the book)
- Combinatorial explosion at large amount of test cases
- No previous solutions to this problem

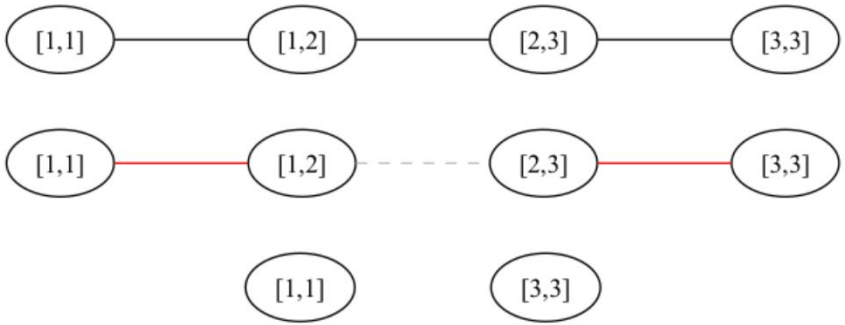


Figure 7: Optimal join

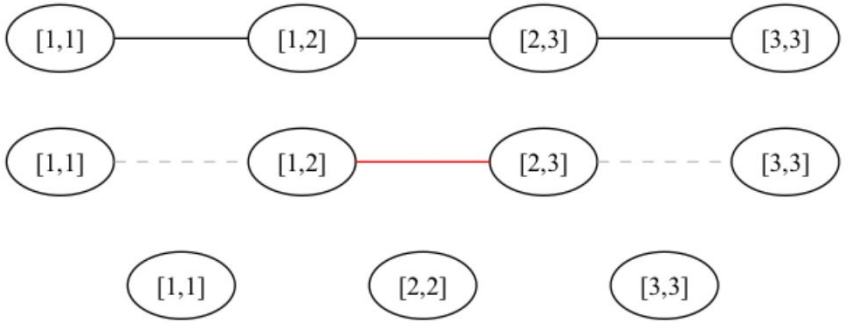


Figure 8: Not optimal join

Graph Reduction Algorithms

- MONKE: Minimal Overhead Node Kollision (Collision) Eliminator
  - Greedy algorithm, starts joining nodes on the first edge it finds in the graph
- LLNR: Least Losing Nodes Reachable
  - Tries to lose the least amount of nodes that could be reached with a BFS
- LLE: Least Losing Edges
  - Tries to lose the least number of edges on a join
- LLC: Least Losing Components
  - Tries to minimise the number of components in the graph
- MLE: Most Losing Edges
  - Tries to lose the most edges on a join

Benchmark results - 1

Number of non-reduced test cases: 14
Number of edges in the initial graph: 39

Algorithm	Runtime	Num. of Test Cases	%
baseline	1ms	14	0%
MONKE	3ms	8	42.86%
MLE	3ms	9	35.71%
LLE	3ms	8	42.86%
LLNR	3ms	8	42.86%
LLC	4ms	8	42.86%

Table 1: price_calculation.gpt benchmark results

Number of non-reduced test cases: 55
Number of edges in the initial graph: 183

Algorithm	Runtime	Num. of Test Cases	%
baseline	3ms	55	0%
MONKE	3ms	22	60%
MLE	5ms	29	47.27%
LLE	9ms	22	60%
LLC	49ms	24	56.36%
LLNR	57ms	22	60%

Table 2: paid_vacation_days.gpt benchmark results

baseline: Generating the test case graph, without any test case reduction

Benchmark results - 2

Number of non-reduced test cases: 328
Number of edges in the initial graph: 11684

Algorithm	Runtime	Num. of Test Cases	%
baseline	0.028s	328	0%
MONKE	0.031s	51	84.45%
MLE	3.11s	71	78.35%
LLE	26.84s	45	86.28%
LLC	130.80s	54	83.54%
LLNR	216.60s	53	83.84%

Table 3: complex_small.gpt benchmark results

Number of non-reduced test cases: 452
Number of edges in the initial graph: 23664

Algorithm	Runtime	Num. of Test Cases	%
baseline	0.066s	452	0%
MONKE	0.084s	69	84.73%
MLE	11.19s	99	78.1%
LLE	111.00s	58	87.17%

Table 4: complex_medium.gpt benchmark results

Benchmark results - 3

Number of non-reduced test cases: 3657
Number of edges in the initial graph: 1229629

Algorithm	Runtime	Num. of Test Cases	%
baseline	47.37s	3657	0%
MONKE	153.00s	354	90.32%

Table 5: complex_hard.gpt benchmark results

Summary

Summary

General Predicate Testing

- Automatic black-box test generation tool
  - Formalization of the GPT Method
- GPT Lang
- Extensions on the original GPT definition
  - Multiintervals
  - Negation, multiple predicates for a variable
  - Disjunction
- Graph reduction
  - MONKE, LLNR, LLE, LLC, MLE
  - Comparison of multiple algorithms

Any questions?