

Automating Differential Testing with Overapproximate Symbolic Execution

Richard Rutledge, Alessandro Orso



2nd UPDATE: Amazon.com Web Site Down For Technical Reasons

June 06, 2008: 04:02 PM EST



(Updated to add information from a company customer-service representative.)

NEW YORK -(Dow Jones)- Amazon.com Inc.'s (AMZN) Web site was down for more than an hour Friday afternoon and remained malfunc

An A... would... the c... didn't... said but

The c... last 1:40 p.m. EDT to 3 p.m. before reappearing with partial functionality.

Dow Jones Newswires employees were still unable to complete a full transaction on the site before getting an error message at the time of this report.

Sponsored Links

Options Investing Streamlined
\$9.95/Option + \$0 per Contract, Any Size. Get Flat Rate Commissions!

An investor's best friend?
In a tough market, lean on Options. The investment with many advantages.

Refinance Now at 5.2% FIXED!
\$200,000 mortgage under \$599/mo. No

[...] the outage was due to an upgrade of the company's Web site [...]

Auto
Over

with
ation

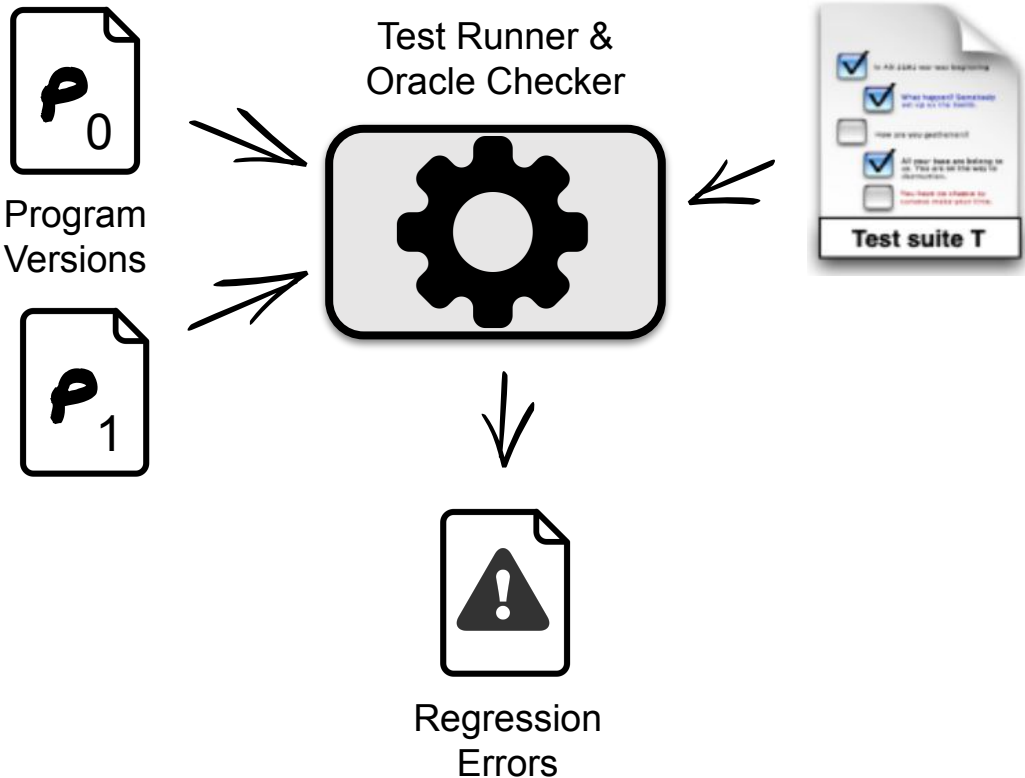
“If only the kernel had a
regression testsuite, everything
would be better.”

-- Greg Kroah-Hartman
keynote on the Linux
kernel at OLS 2006



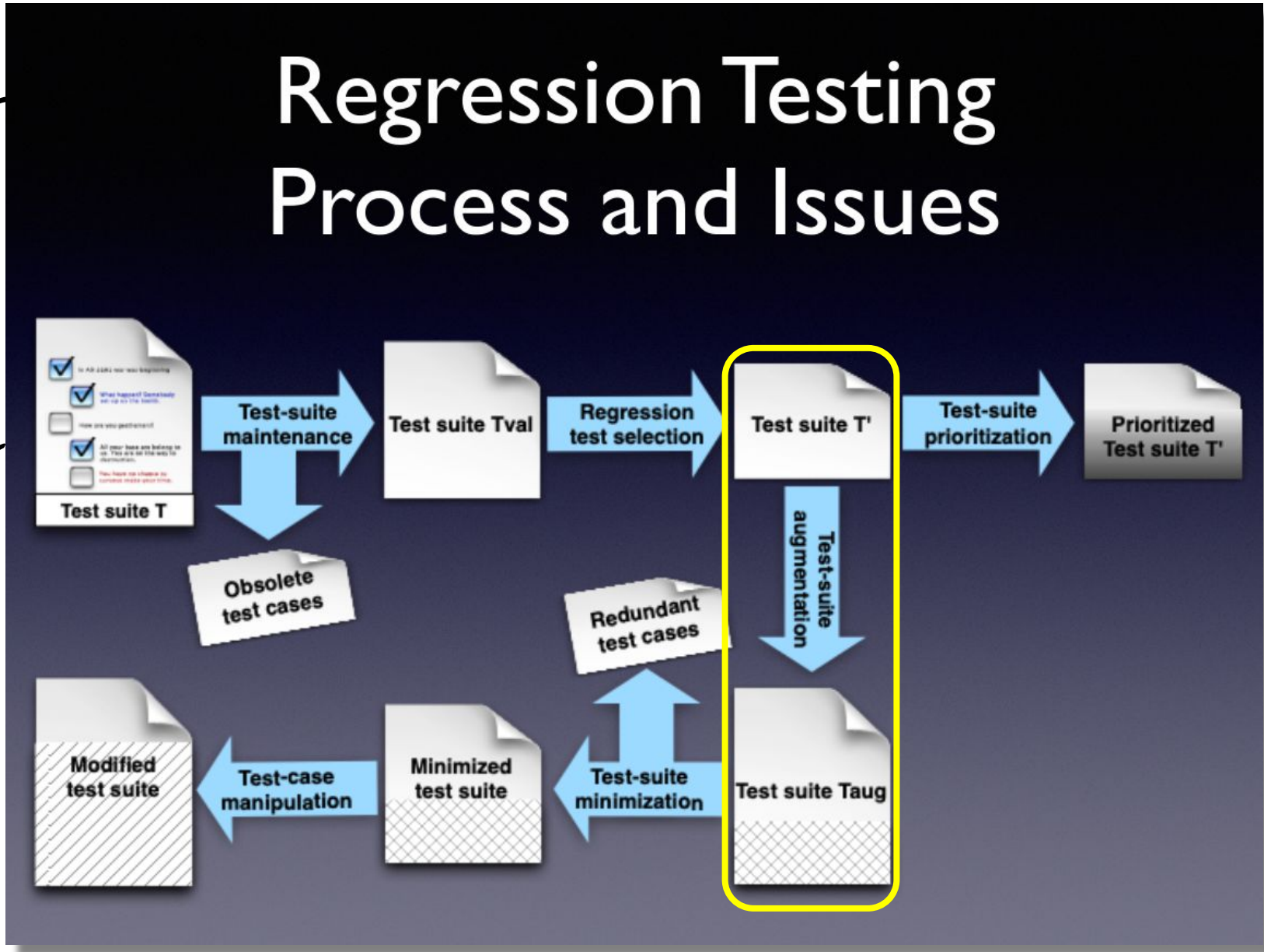
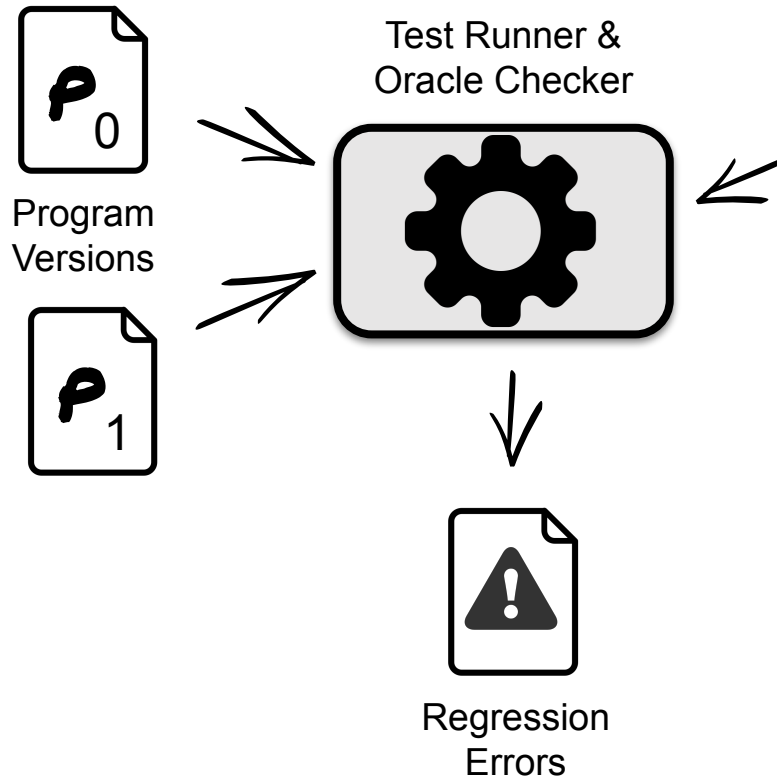


Regression Testing

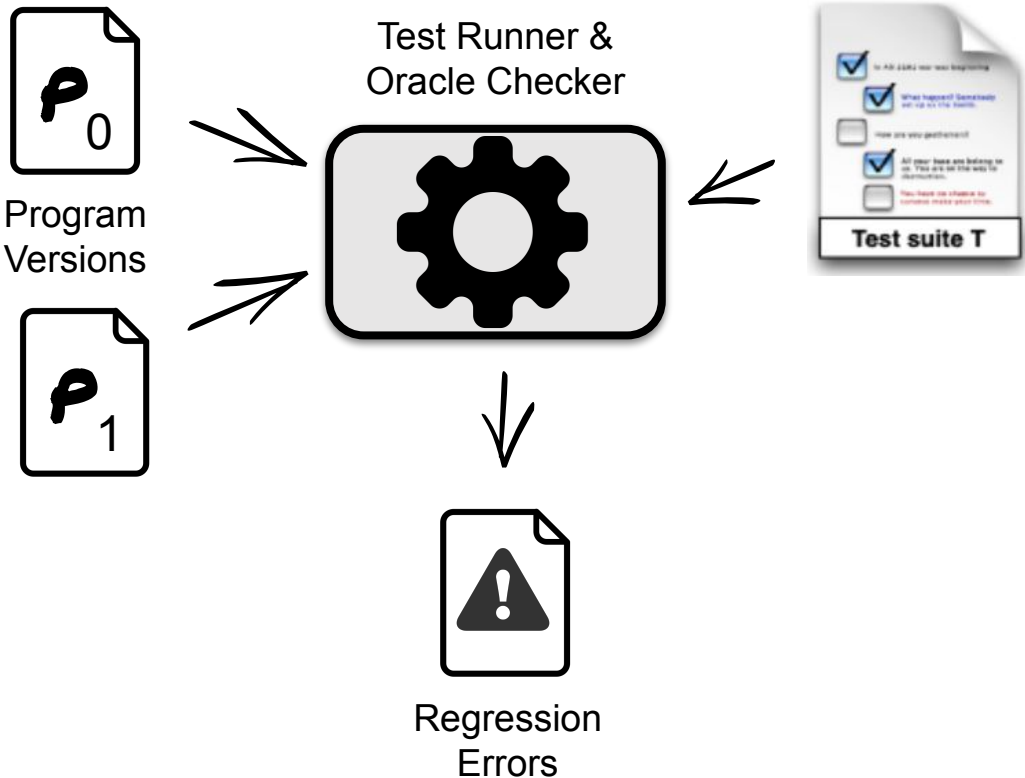


Regression Testing

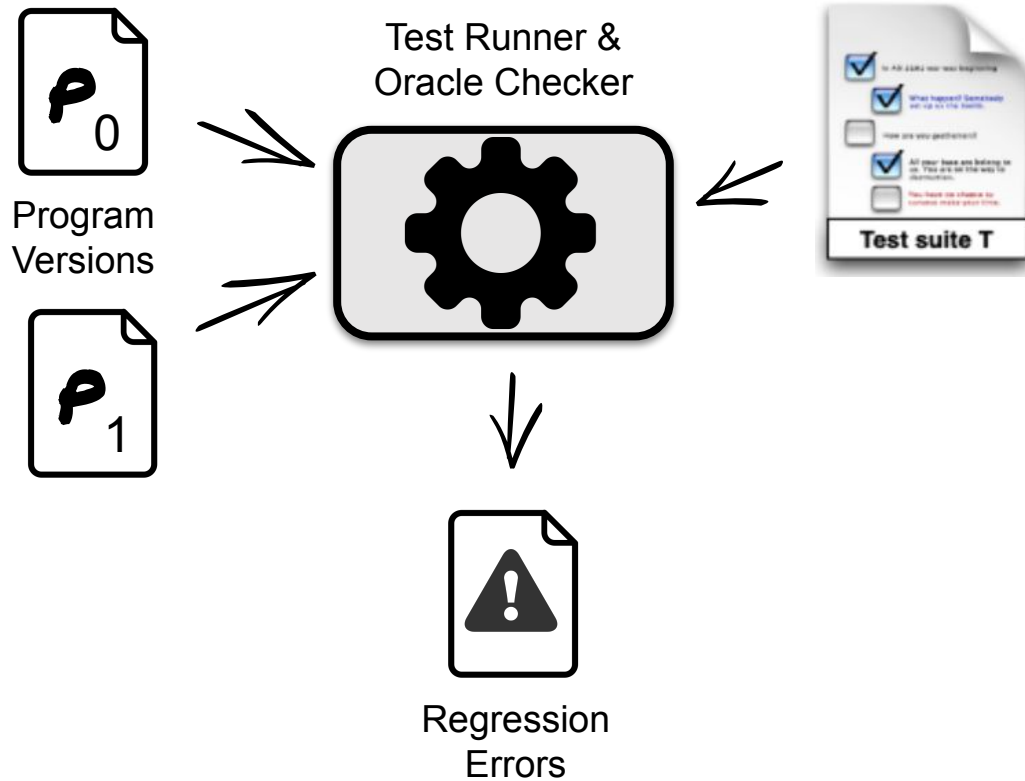
Regression Testing Process and Issues



Regression Testing



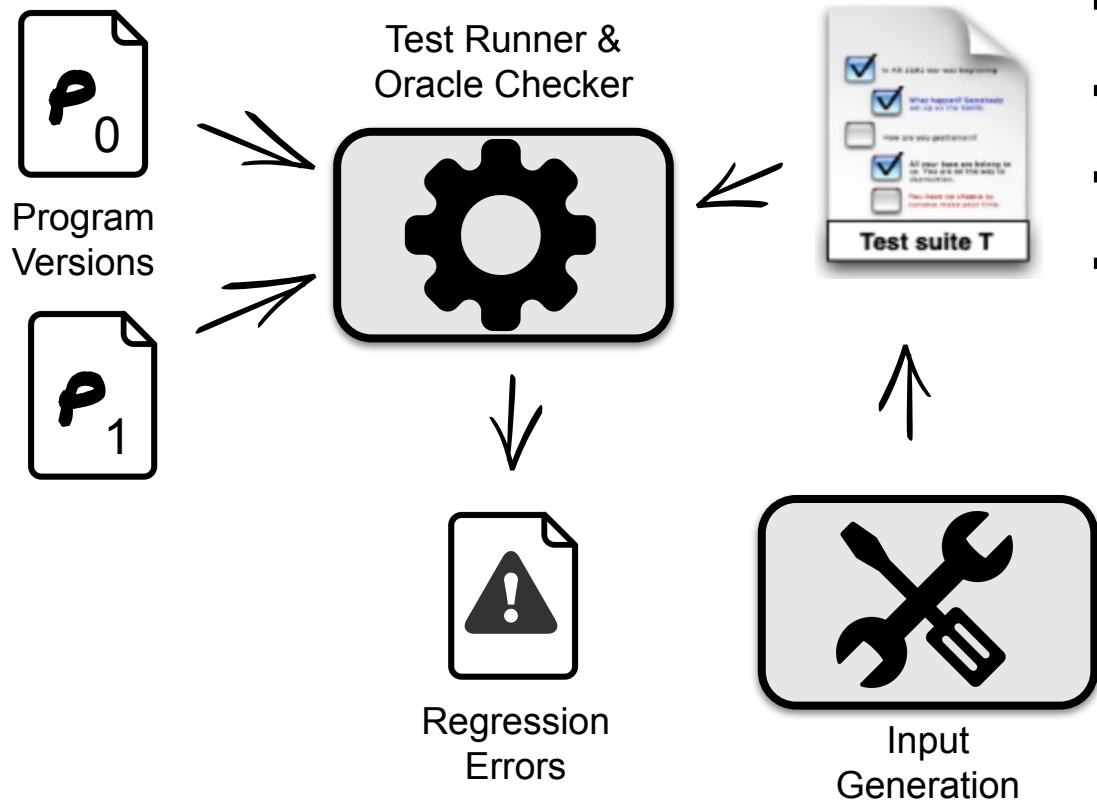
Regression Testing



Issues with regression test suites

- Focus on core behavior
- Provide limited coverage
- Use approximated oracles
- Sometimes not present at all

First "Intuition": Input Generation

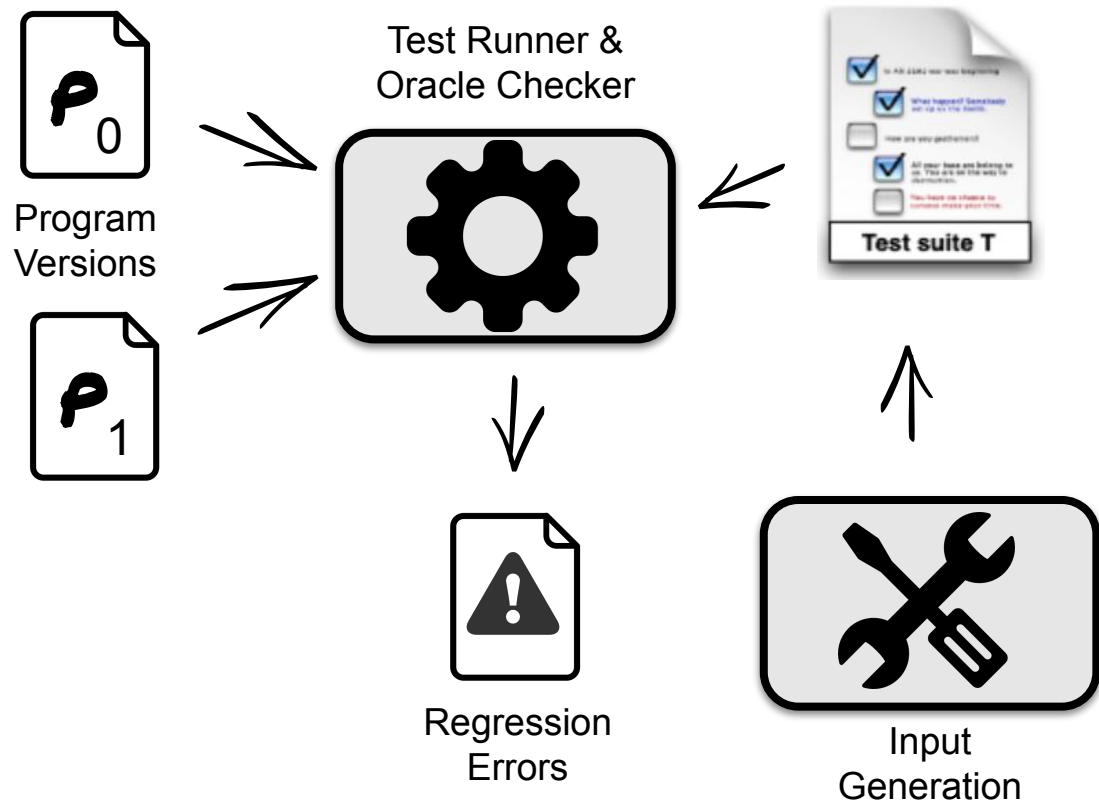


Issues with regression test suites

- Focus on core behavior
- Provide limited coverage
- Use approximated oracles
- Sometimes not present at all

- Inherent limitations
- Oracle problem

Second Intuition: Limited Scope and Differential Testing

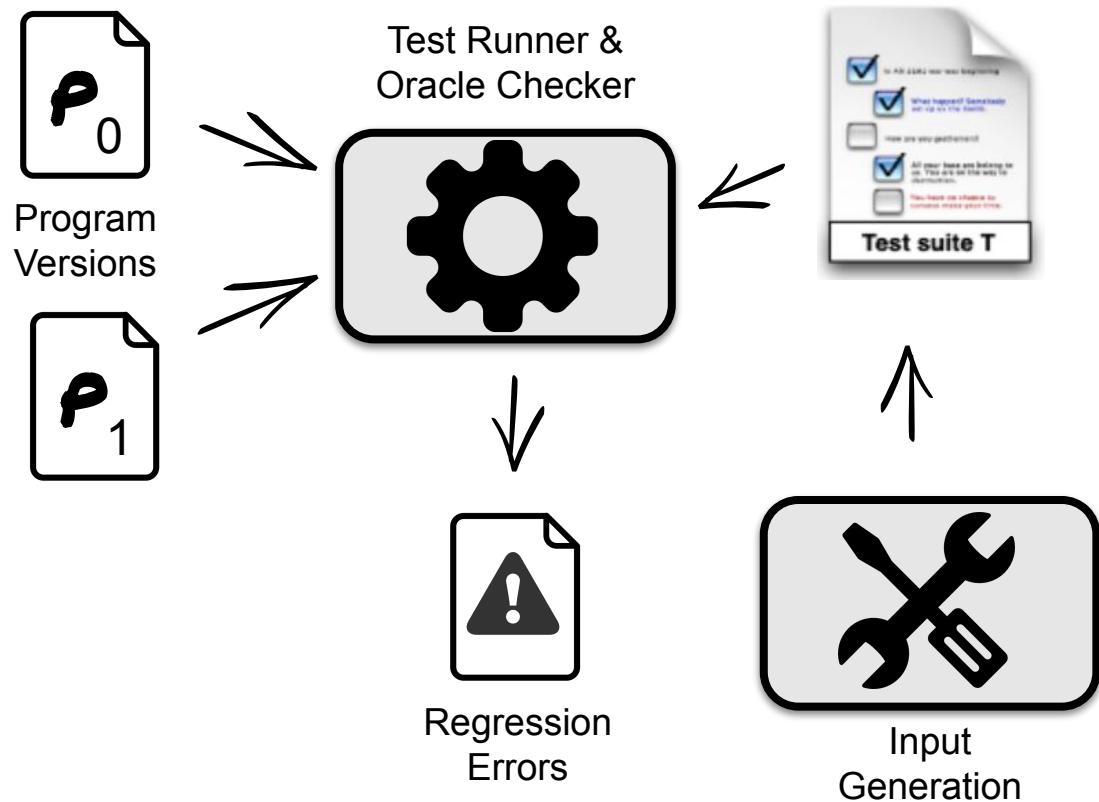


Focused input generation
+

P_0 as oracle

- focus on changed code
- thorough coverage
- no need for oracles

Second Intuition: Limited Scope and Differential Testing



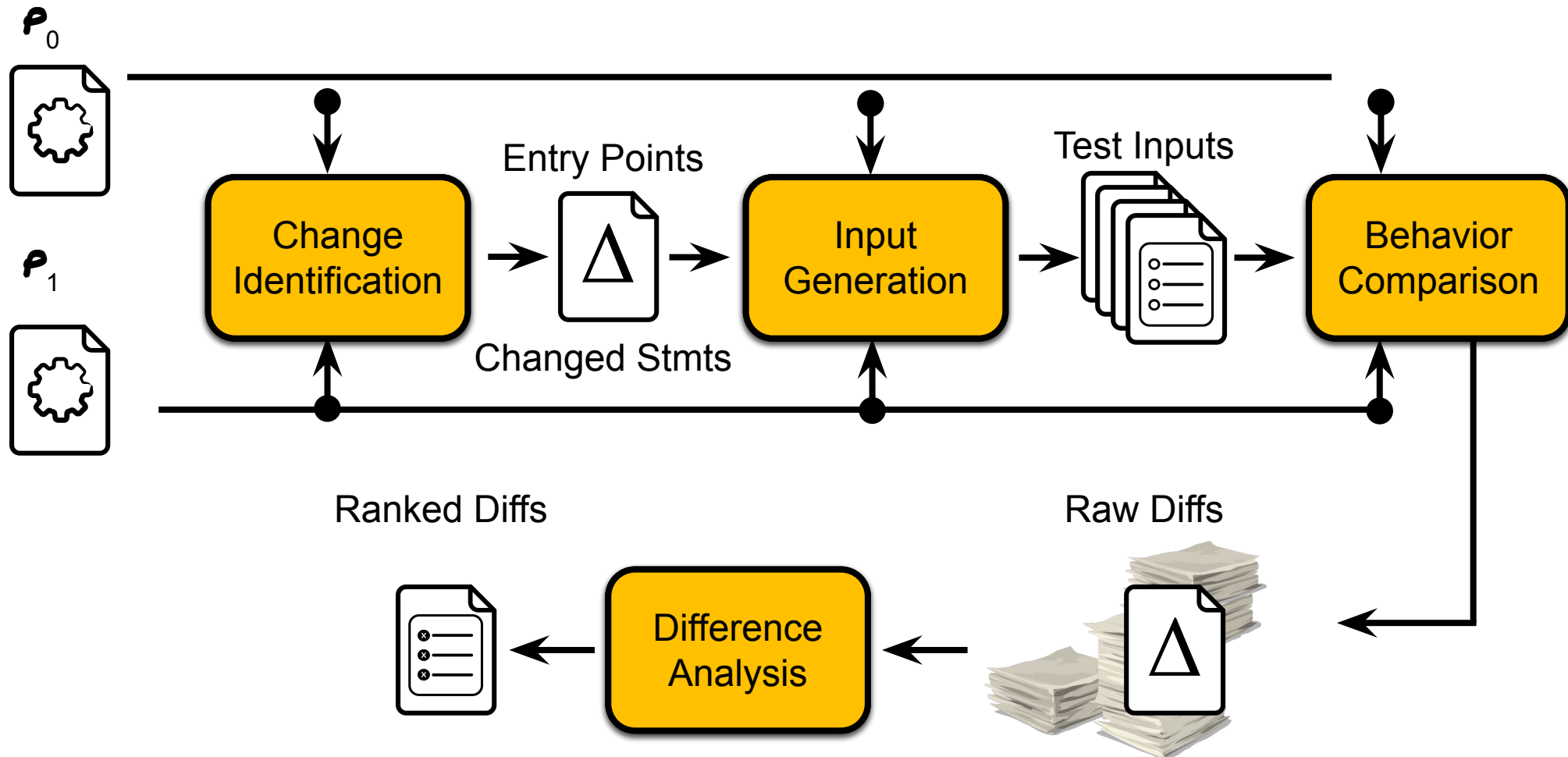
Focused input generation
+

P_0 as oracle

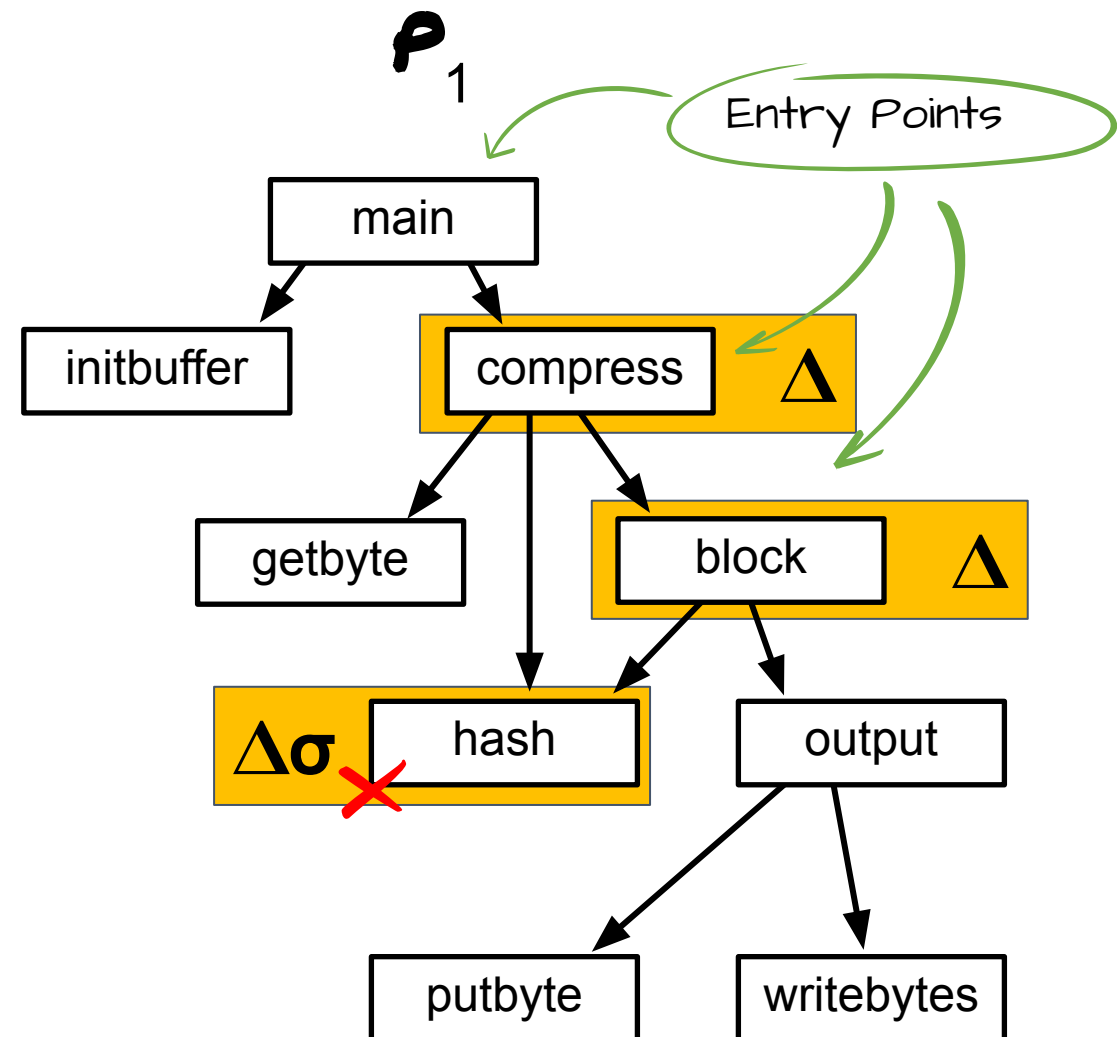
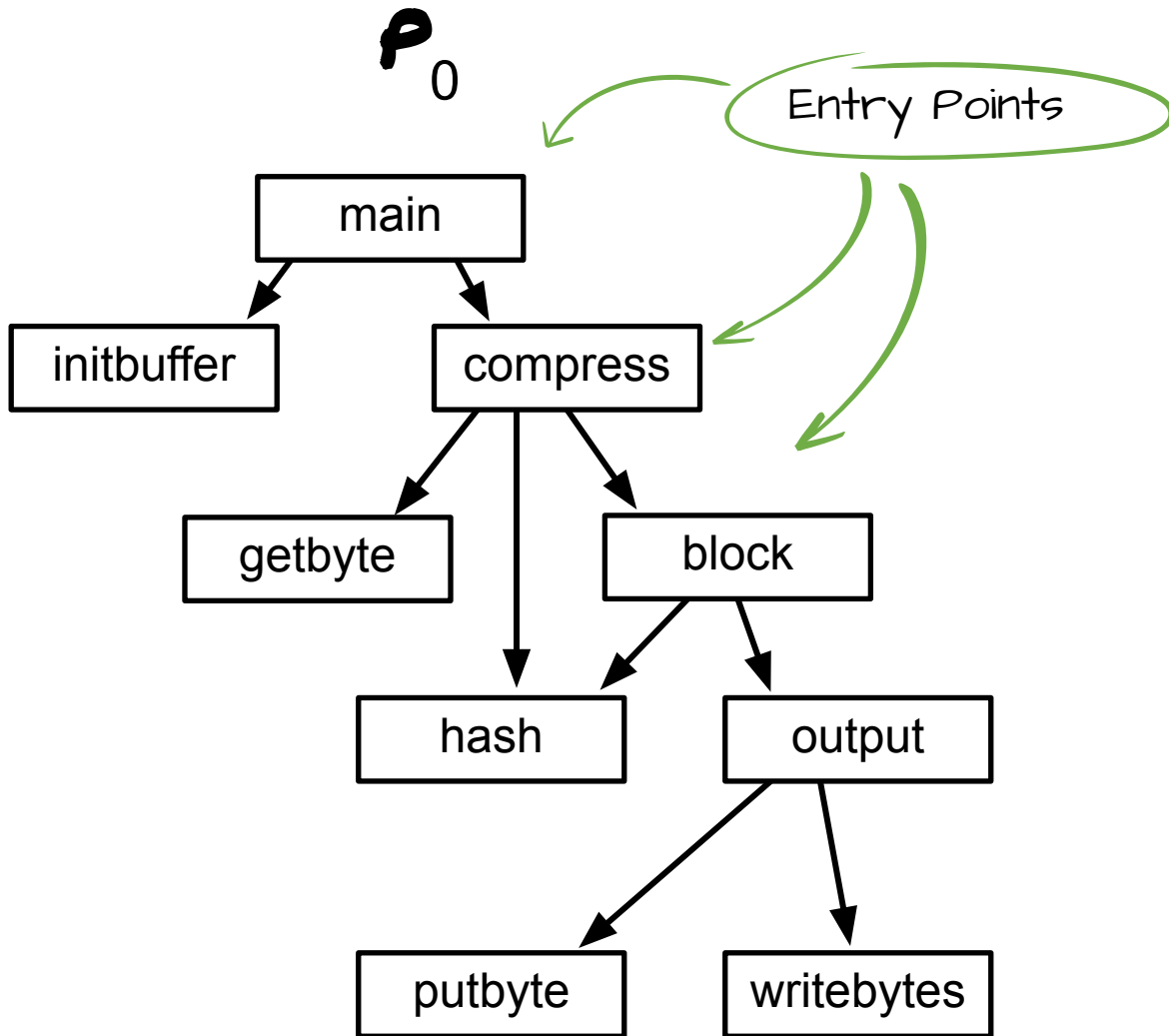
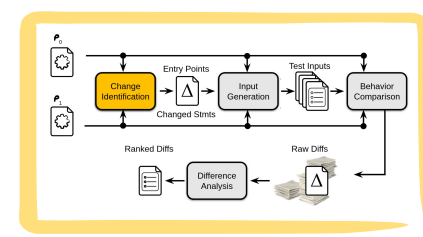
- focus on changed code
- thorough coverage
- no need for oracles

Usage scenario: frequent use
(e.g., every save or before
merge into release branch)

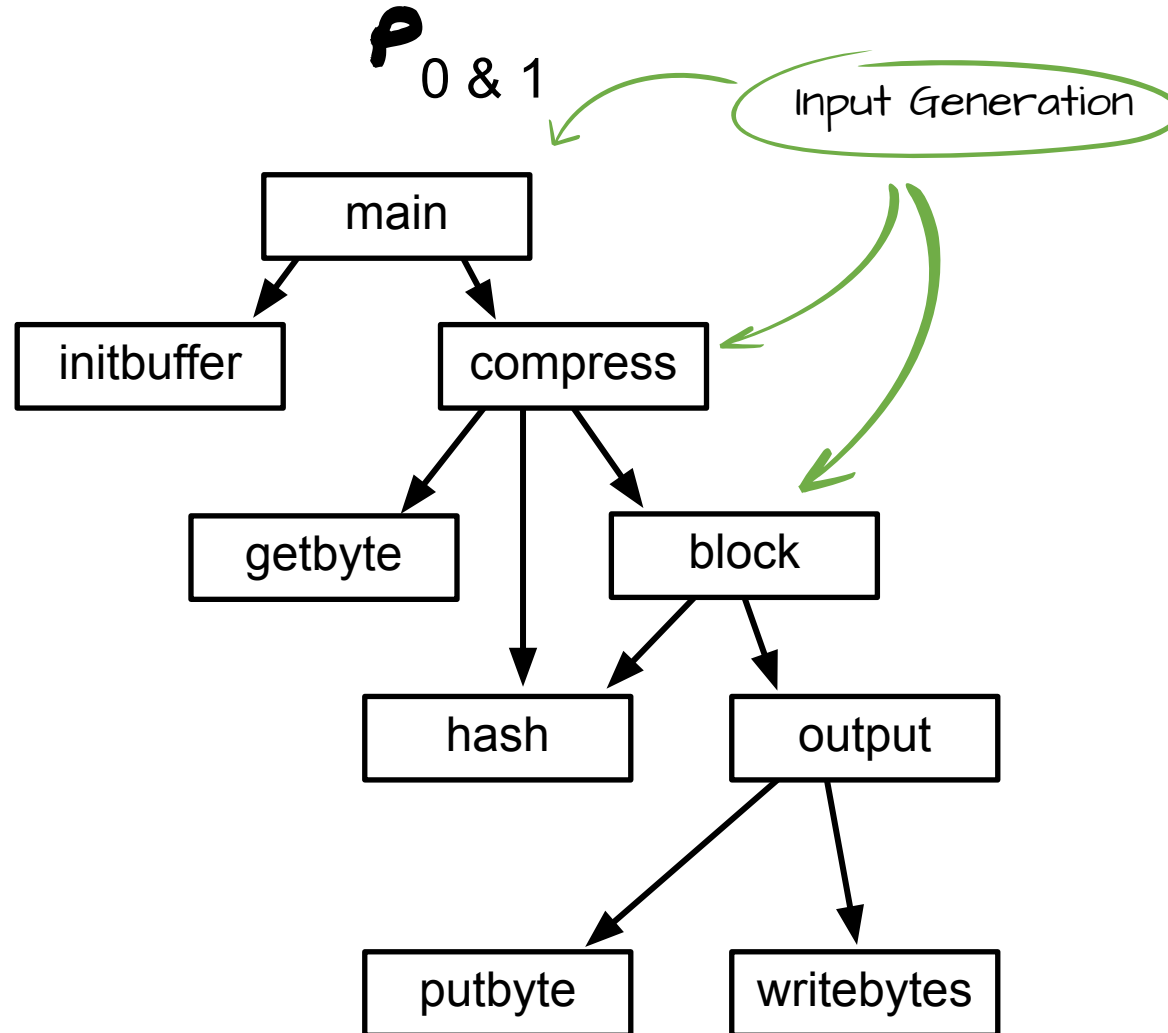
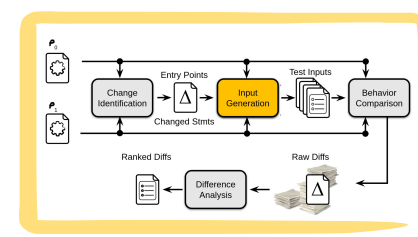
Overapproximate Differential Regression Testing (ODIT) Overview

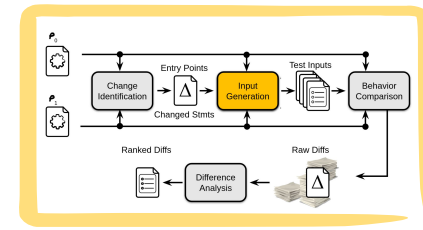


Change Identification



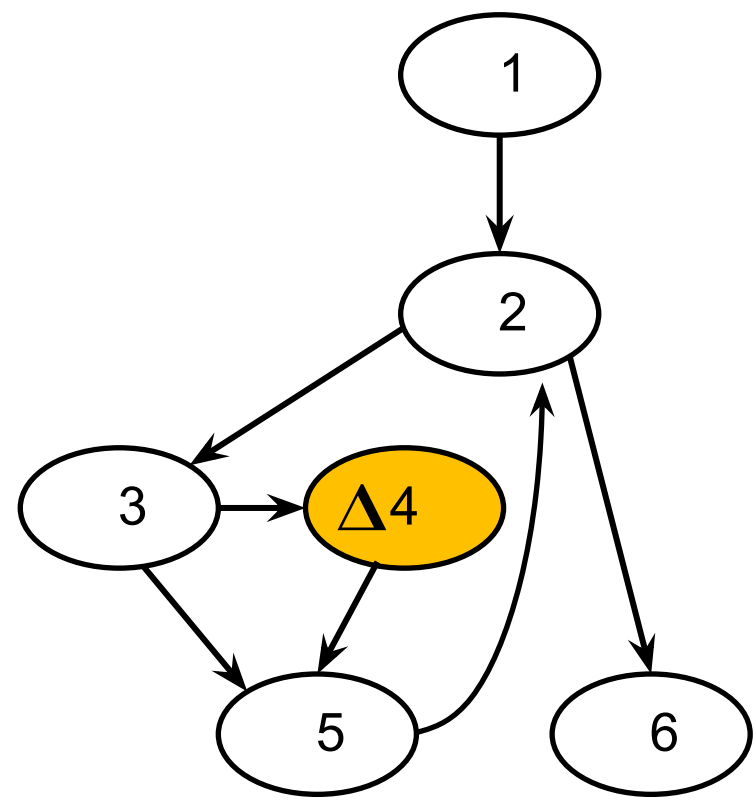
Input Generation: Underconstrained SymEx





Input Generation: Input Selection

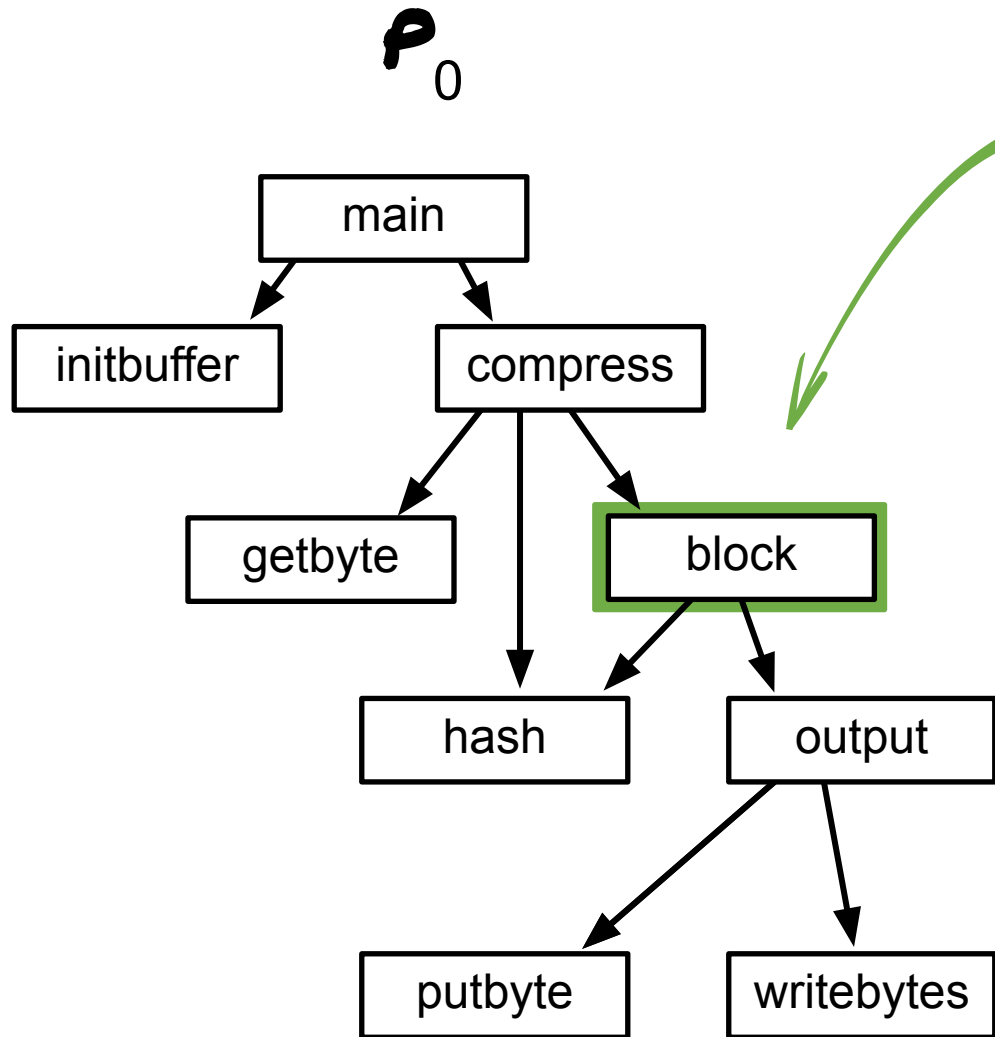
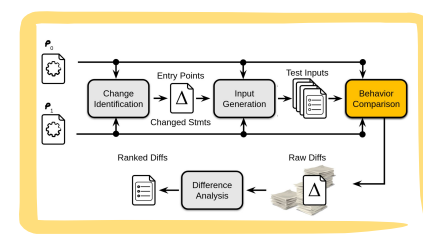
block Control Flow Graph (CFG)



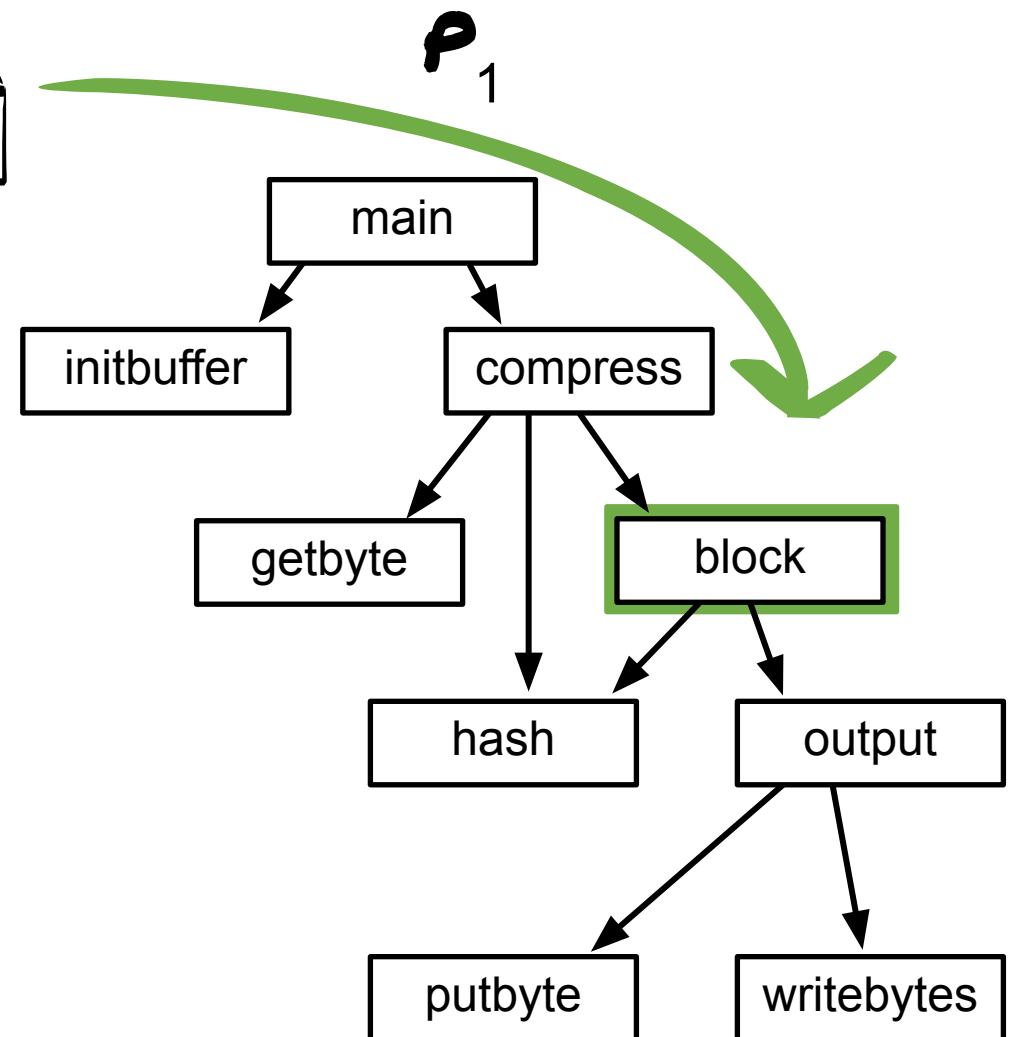
Input execution paths:

- ~~1, 2, 3, 5, 2, 6~~ ❌
- 1, 2, 3, 4, 5, 2, 6 ➤ 📄
- 1, 2, 3, 5, 2, 3, 4, 5, 2, 6 ➤ 📄
- ~~1, 2, 6~~ ❌

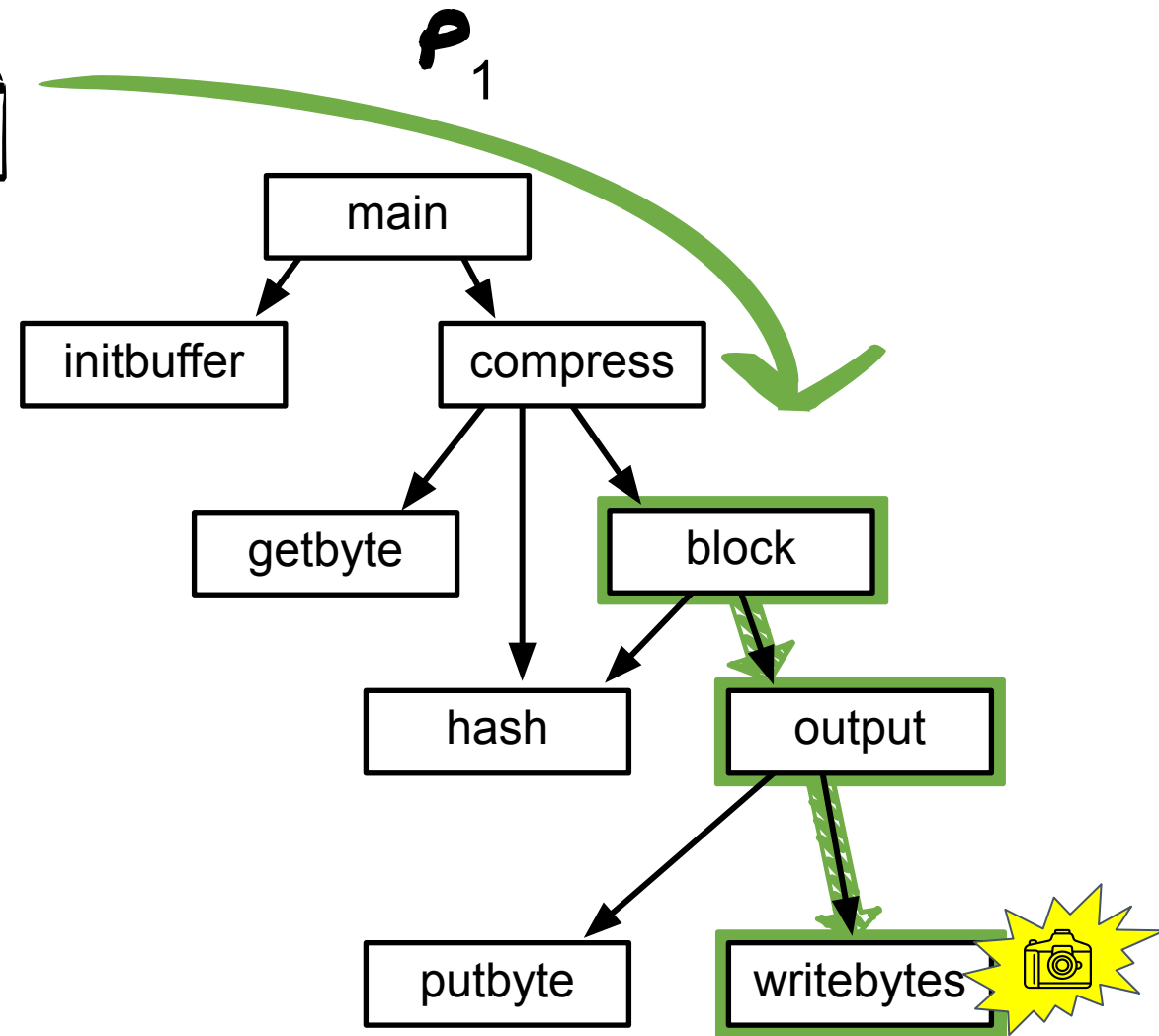
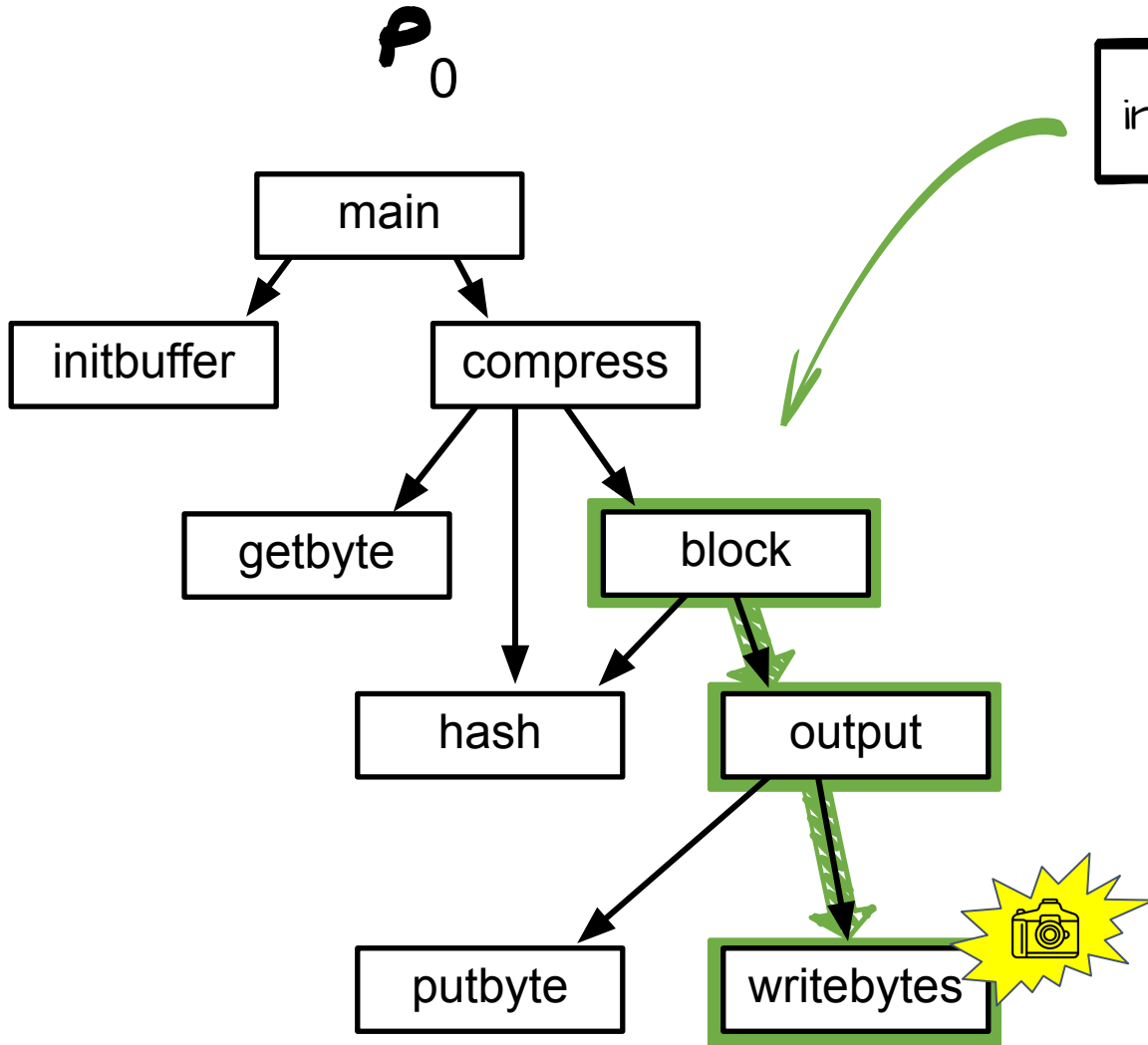
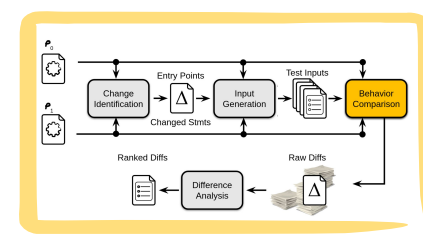
Behavior Comparison: Replay



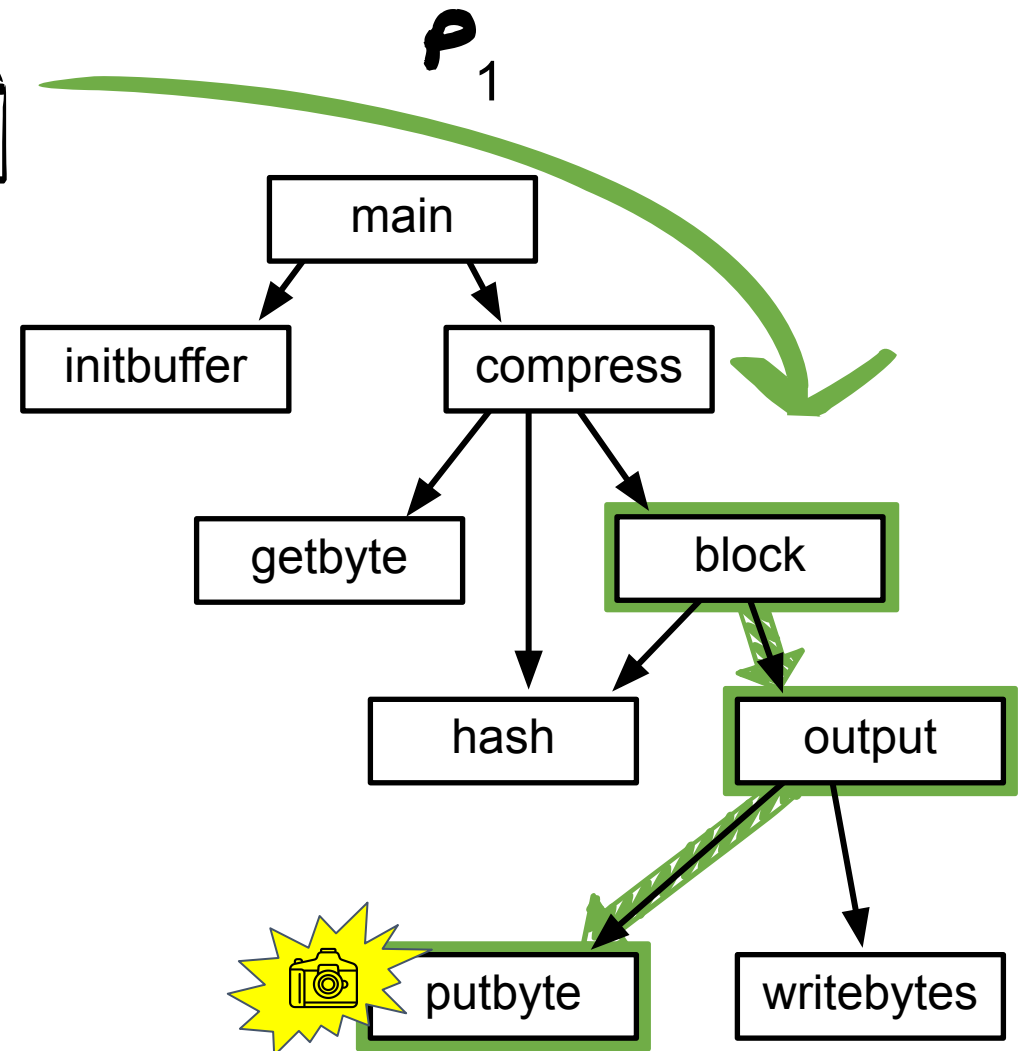
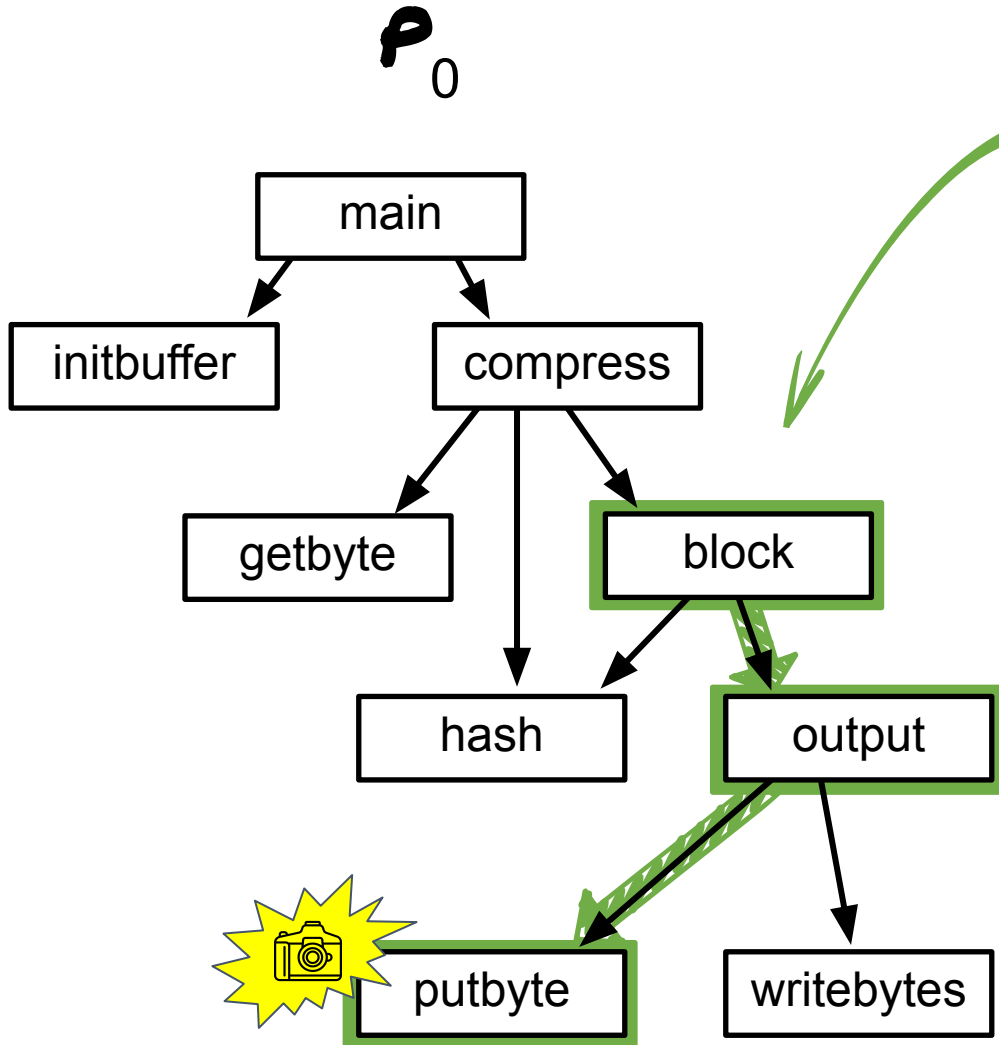
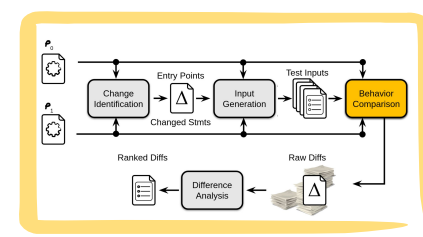
input



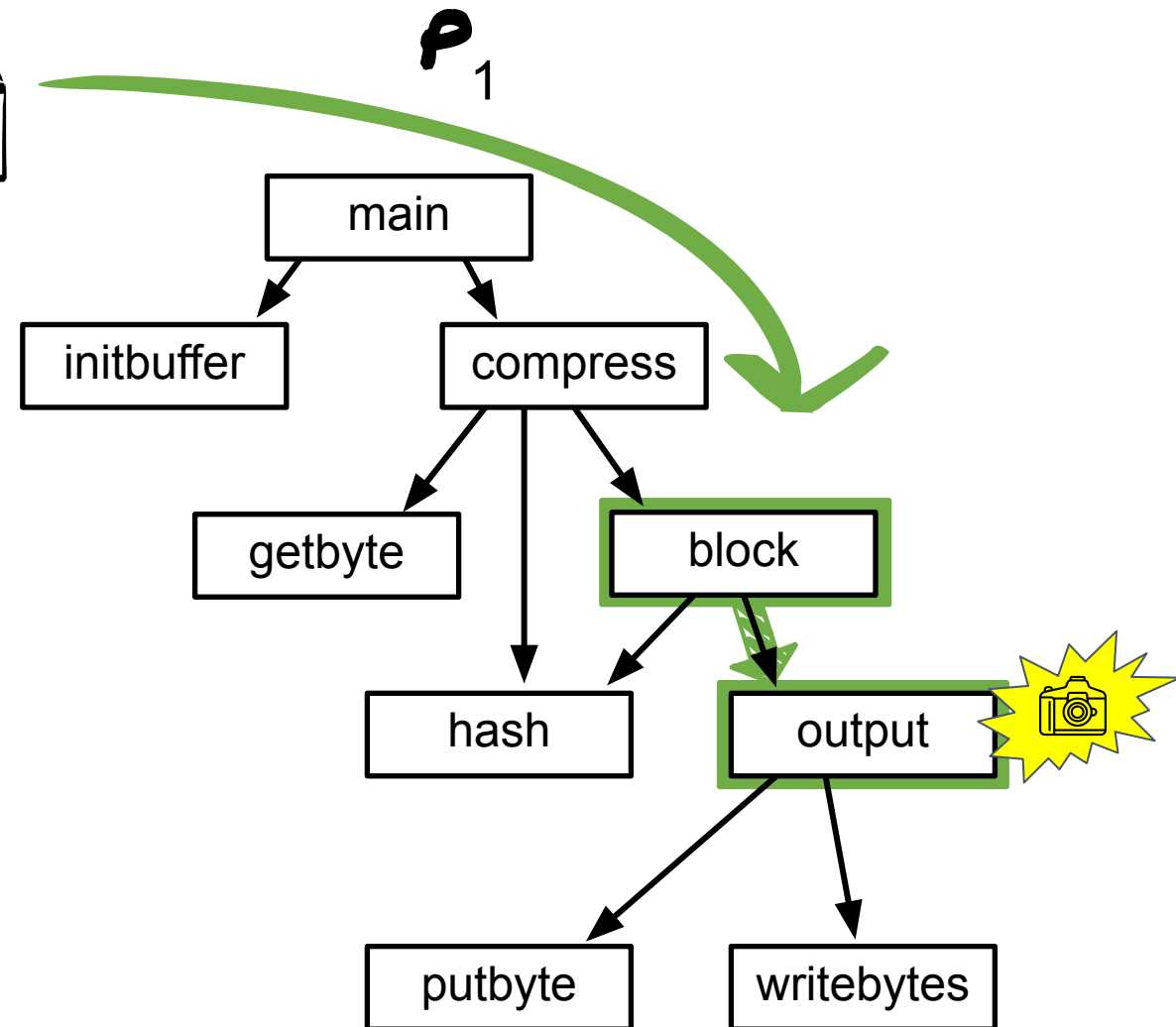
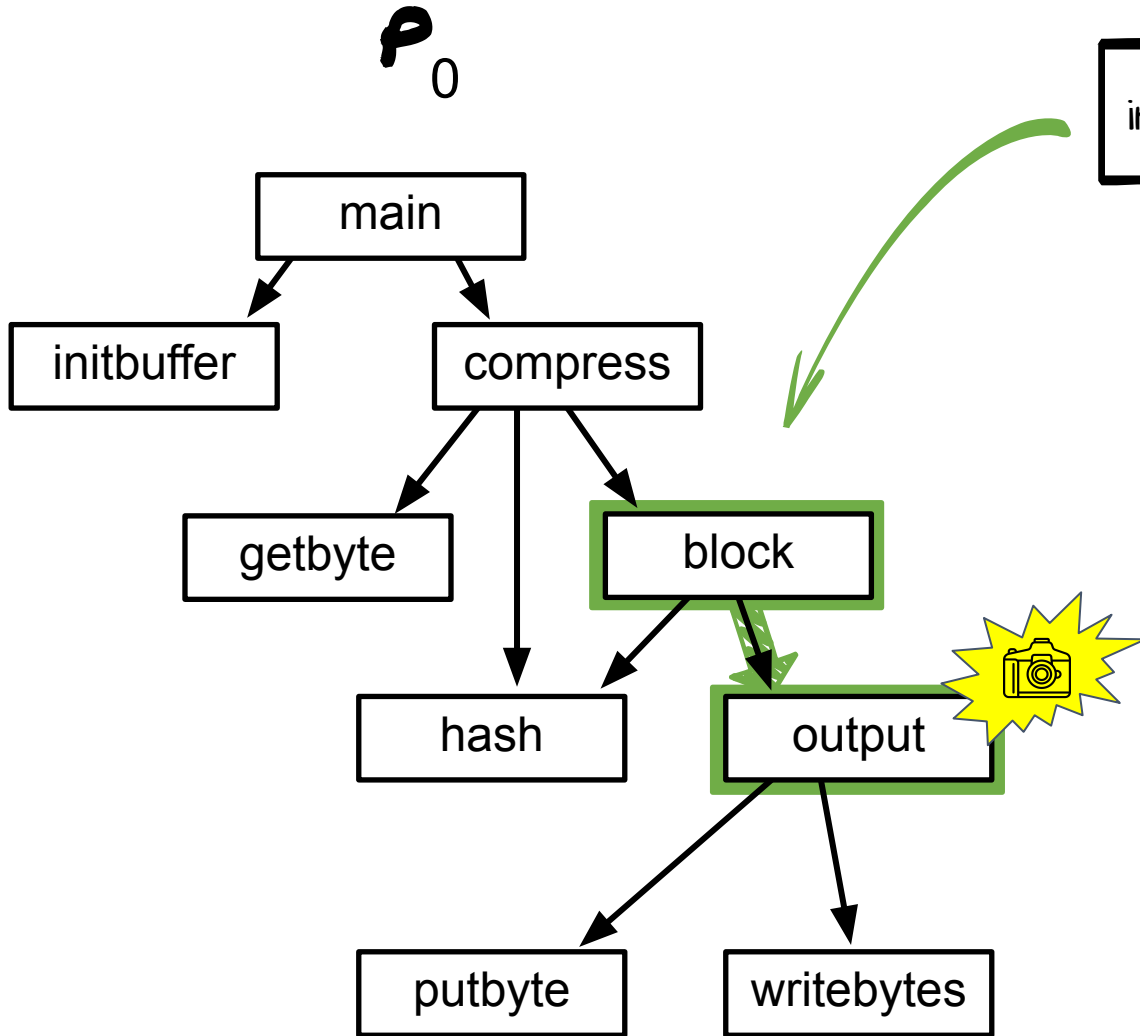
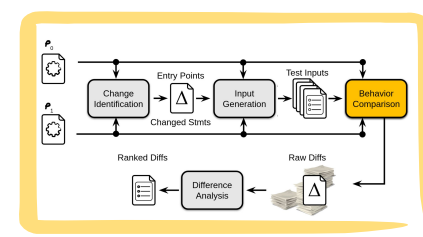
Behavior Comparison: Replay



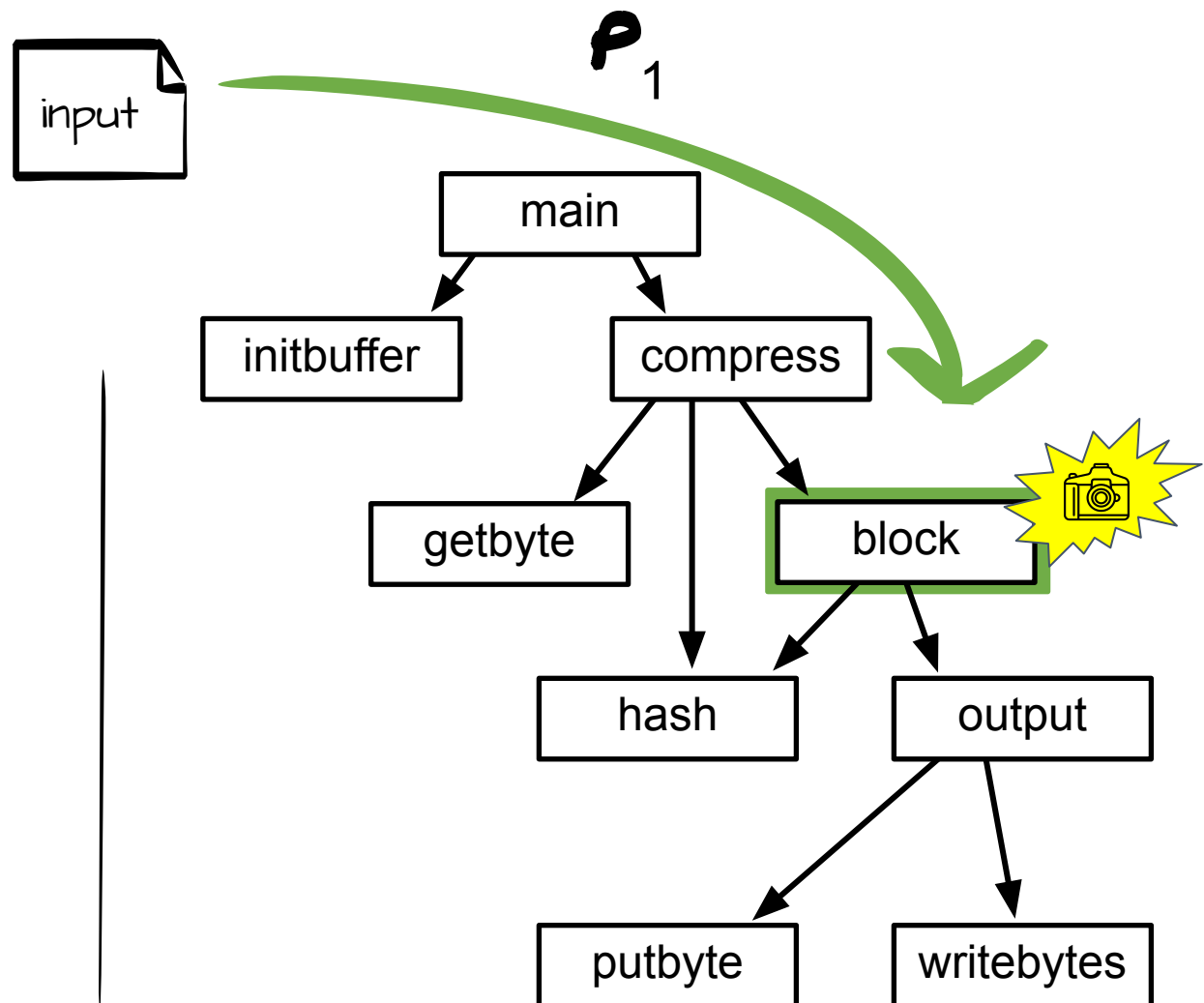
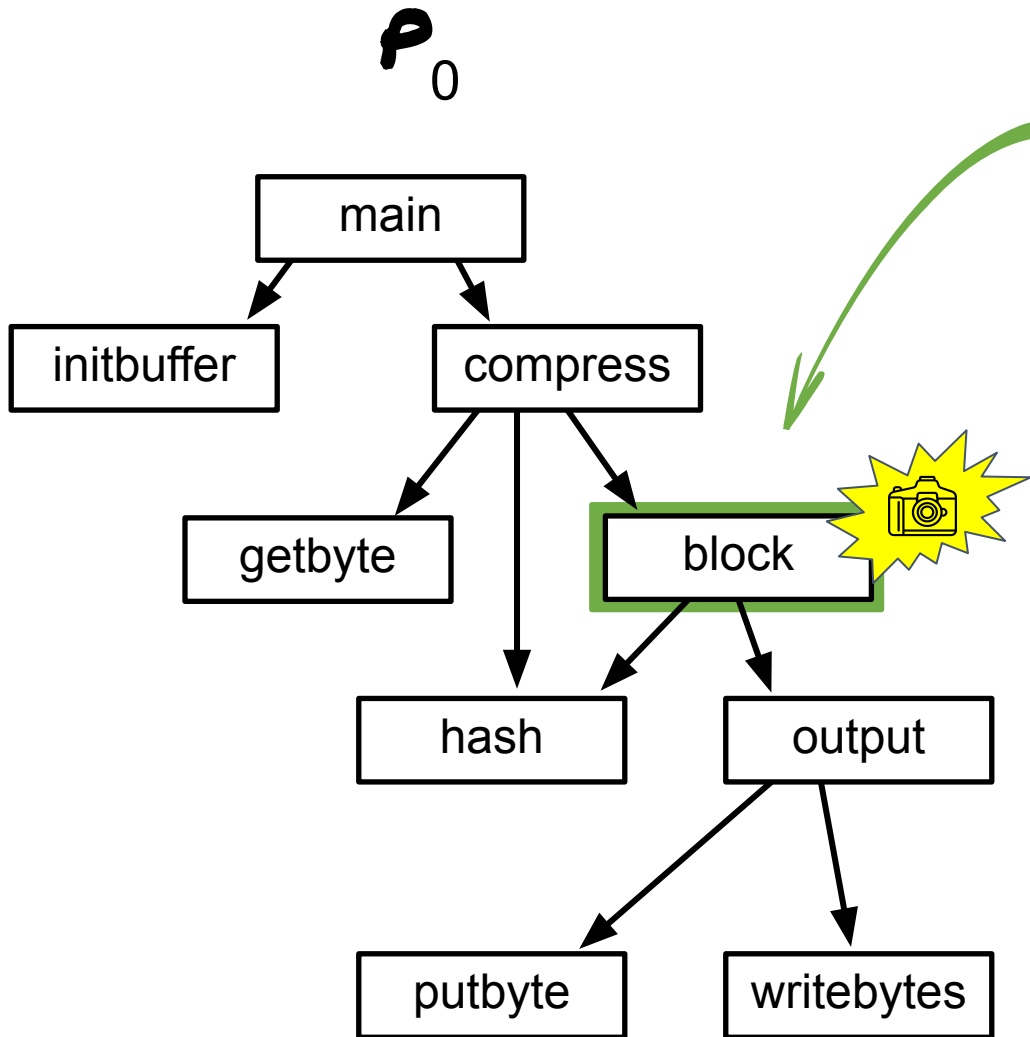
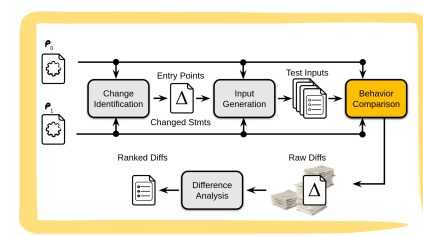
Behavior Comparison: Replay

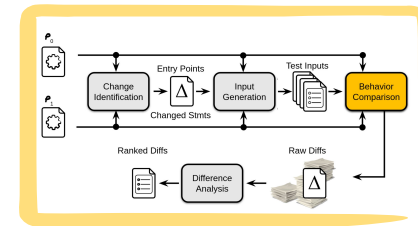


Behavior Comparison: Replay

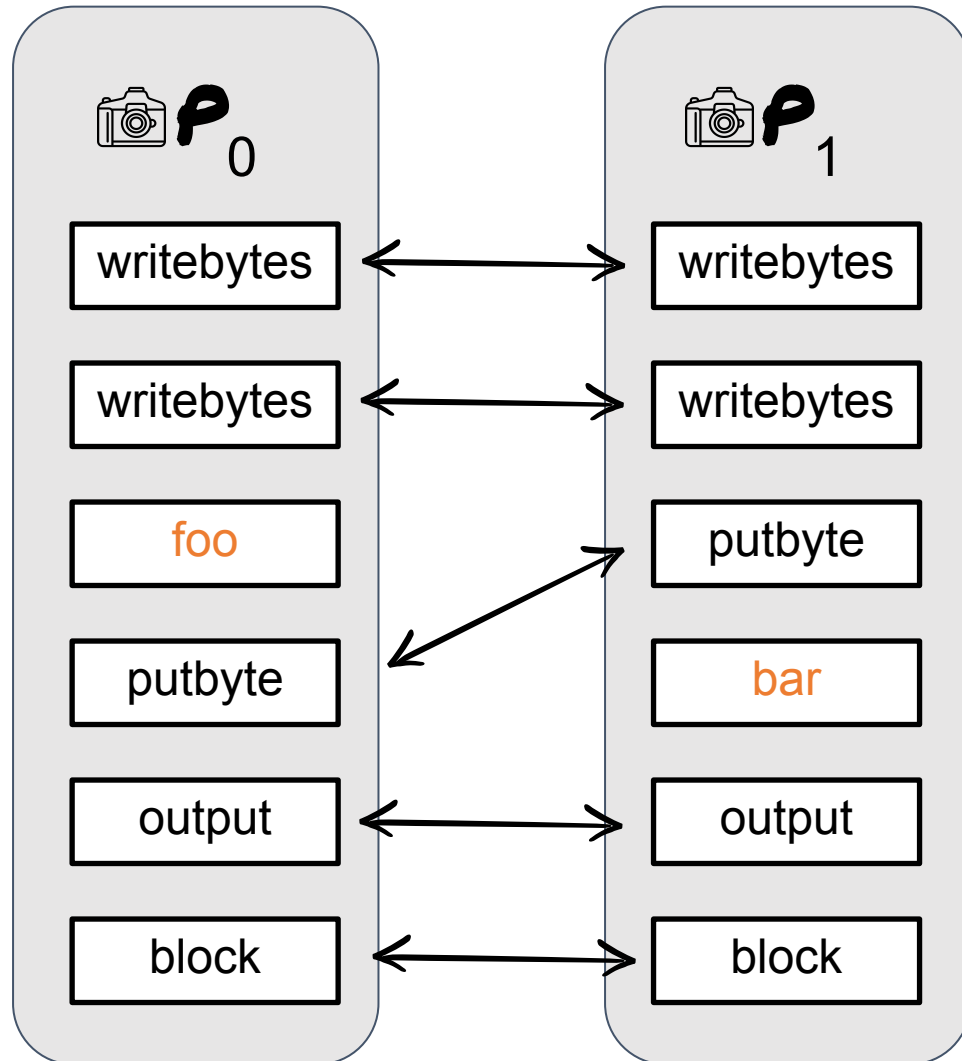


Behavior Comparison: Replay





Behavior Comparison: Alignment and State



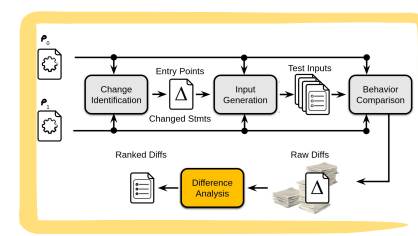
Alignment

- algorithm based on longest common subsequences

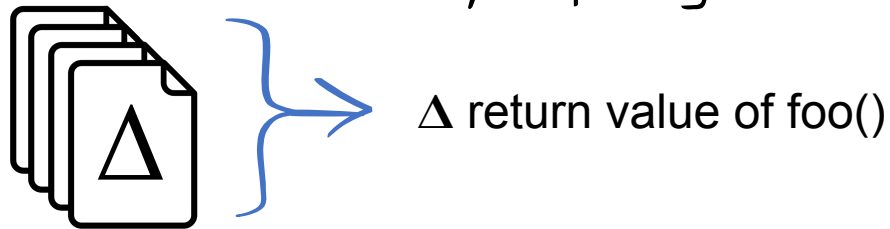
Address space elements compared:

- termination
- returning function value
- global variables
- output streams
- output parameters

Difference Analysis



- Group differences by Δ program elements



- Dependent differences based on co-occurrence

$\Delta x \rightarrow \Delta \text{ret foo}()$

```
int x;  
int bar() {  
    x = foo();  
}
```

root difference = $\Delta \text{ret foo}()$

- Rank by distance from changed code in the dynamic call sequence.

foo() modified

1: foo() $\rightarrow \Delta x$

2: foo(), same₁(), same₂(), same₃(), ... , same_N() $\rightarrow \Delta y$

Rank

1. Δy

2. Δx

Evaluation: Implementation and Research Questions

Implementation

- Program analysis and differencing: clang & llvm
- Symbolic execution engine: forked from KLEE 1.3

Research Questions

- RQ1: Can ODiT detect and effectively rank regressions?
- RQ2: How does ODiT, which overapproximates behavior, compare to a tool that underapproximates behavior?
- RQ3: How does ODiT perform on refactored code?

Evaluation: Setup and Benchmarks

RQ1

- CoREBench: coreutils, find, and grep
 - Bug oracles to compute TPs and FPs
-

RQ3

- Redis

program	CoREBench IDs	LOC
rm	1	1044
cut	3, 6, 12, 17, 21	519
tail	4, 5, 16	1039
seq	7, 8, 9, 18, 19, 20	254
cp	10	2498
ls	13, 14	3106
du	15	624
expr	22	583
find	23 - 37	8,738
grep	38 - 52	6,153
redis	N/A	121,989

CoREBench Selection

We considered the 70 CoREBench regressions, omitting those

- requiring 32-bit compilation
- using unsupported multibyte locales
- redundantly resulting from the same defect

=> this eliminated 9 of the 70 regressions)

We then analyzed the remaining 61 regressions to identify those for which we could reliably define a ground truth (bug oracle)

=> 43 regressions

RQ1: Detection and Ranking

benchmark	inputs	diffs	TP	FP	precision	rank
01-rm	671	0	0	0	-	N/A
03-cut	30641	5	0	5	0.0 %	N/A
04-tail	11407	1	1	0	100.0 %	1
05-tail	8311	1	0	1	0.0 %	N/A
06-cut	3198	5	2	3	40.0 %	1
07-seq	13427	2	1	1	50.0 %	1
08-seq	14088	3	1	2	33.3 %	3
09-seq	15248	2	2	0	100.0 %	1
10-cp	4239	2	2	0	100.0 %	1
12-cut	2012	2	2	0	100.0 %	1
13-ls	4583	8	5	3	62.5 %	2
14-ls	27704	51	0	51	0.0 %	N/A
15-du	2965	15	13	2	86.7 %	1
16-tail	586	0	0	0	-	N/A
17-cut	3142	1	0	1	0.0 %	N/A
18-seq	9069	5	3	2	60.0 %	2
19-seq	25758	22	0	22	0.0 %	N/A
20-seq	25918	16	0	16	0.0 %	N/A
21-cut	58	0	0	0	-	N/A
22-expr	168	13	0	13	0.0 %	N/A

benchmark	inputs	diffs	TP	FP	precision	rank
23-find	2552	67	1	66	1.5 %	67
24-find	22994	3	0	3	0.0 %	N/A
26-find	180975	65	10	55	15.4 %	1
27-find	7771	1	0	1	0.0 %	N/A
28-find	89420	4	0	4	0.0 %	N/A
30-find	35945	1	1	0	100.0 %	1
31-find	2012	2	2	0	100.0 %	1
32-find	4583	8	5	3	62.5 %	2
33-find	27704	51	0	51	0.0 %	N/A
34-find	2965	15	13	2	86.7 %	1
35-find	586	0	0	0	-	N/A
36-find	3142	1	0	1	0.0 %	N/A
37-find	9069	5	3	2	60.0 %	2
38-find	25758	22	0	22	0.0 %	N/A
39-find	25918	16	0	16	0.0 %	N/A
40-find	58	0	0	0	-	N/A
41-grep	168	13	0	13	0.0 %	N/A
42-grep	2012	3	3	0	100.0 %	1



- In 58% of the cases, actual regressions in Top-3 (often Top-1)
- In 47% of the cases, FPs ranked higher than TPs
- In 33% of the cases, no false positives

RQ1: Detection and Ranking

benchmark	inputs	diffs	TP	FP	precision	rank
01-rm	671	0	0	0	-	N/A
03-cut	30641	5	0	5	0.0 %	N/A
04-tail	11407	1	1	0	100.0 %	1
05-tail	8311	1	0	1	0.0 %	N/A
06-cut	3198	5	2	3	40.0 %	1
07-seq	13427	2	1	1	50.0 %	1
08-seq	14088	3	1	2	33.3 %	2
09-seq	15248	2	2	0	100.0 %	1
10-cp	4239	0	0	0	-	N/A
12-cut	28606	7	3	4	42.9 %	3
13-ls	13062	3	2	1	66.7 %	1
14-ls	10186	10	10	0	100.0 %	1
15-du	1402	10	8	2	80.0 %	1
16-tail	8296	1	0	1	0.0 %	N/A
17-cut	28573	7	3	4	42.9 %	3
18-seq	15248	2	2	0	100.0 %	1
19-seq	8533	3	1	2	33.3 %	3
20-seq	15250	2	2	0	100.0 %	1
21-cut	18841	11	11	0	100.0 %	1
22-expr	2644	1	1	0	100.0 %	1

benchmark	inputs	diffs	TP	FP	precision	rank
23-find	2552	67	1	66	1.5 %	67
24-find	22994	3	0	3	0.0 %	N/A
26-find	180975	65	10	55	15.4 %	1
27-find	7771	1	0	1	0.0 %	N/A
28-find	89420	4	0	4	0.0 %	N/A
30-find	35945	7	7	0	100.0 %	1
31-find	180873	64	10	54	15.6 %	1
32-find	52459	9	2	7	22.2 %	1
33-find	52268	9	2	7	22.2 %	1
34-find	34835	7	0	7	0.0 %	N/A
36-find	68074	4	0	4	0.0 %	N/A
37-find	89012	2	2	0	100.0 %	1
38-grep	4583	8	5	3	62.5 %	2
41-grep	27704	51	0	51	0.0 %	N/A
42-grep	2965	15	13	2	86.7 %	1
44-grep	586	0	0	0	-	N/A
45-grep	3142	1	0	1	0.0 %	N/A
46-grep	9069	5	3	2	60.0 %	2
47-grep	25758	22	0	22	0.0 %	N/A
48-grep	25918	16	0	16	0.0 %	N/A
49-grep	58	0	0	0	-	N/A
51-grep	168	13	0	13	0.0 %	N/A
52-grep	2012	3	3	0	100.0 %	1

RQ2: Compare with Shadow

benchmark	ODiT	Shadow
01-rm	x	x
03-cut	x	x
04-tail	✓	x
05-tail	✓	✓
06-cut	✓	✓
07-cut	✓	✓
08-cut	✓	✓
09-cut	x	x
10-cp	x	✓
12-cut	✓	✓

benchmark	ODiT	Shadow
13-ls	✓	✓
14-cut	✓	✓
15-cut	✓	✓
16-cut	✓	✓
17-cut	✓	✓
18-seq	✓	x
19-seq	✓	x
20-seq	✓	x
21-cut	✓	✓
22-expr	✓	x



- ODiT detected about twice as many regressions as Shadow.
- But, as cost of potential false-positives.

RQ2: Compare with Shadow

benchmark	ODiT	Shadow
01-rm	x	x
03-cut	x	x
04-tail	✓	x
05-tail	x	✓
06-cut	✓	✓
07-seq	✓	x
08-seq	✓	x
09-seq	x	x
10-cp	x	✓
12-cut	✓	✓

benchmark	ODiT	Shadow
13-ls	✓	✓
14-ls	✓	x
15-du	✓	x
16-tail	x	✓
17-cut	✓	✓
18-seq	✓	x
19-seq	✓	x
20-seq	✓	x
21-cut	✓	✓
22-expr	✓	x

RQ3: Refactored Code

- Collected all commits to Redis 5.0.0–5.0.1
- Eliminated irrelevant commits (no changes to relevant source code)
- Manually categorized remaining 53 commits

category	count	inputs	diffs
comment	1	0	0
data	1	0	0
api	1	0	0
behavior	23	N/A	0
refactoring	10	7852	0



ODiT produced no false-positives for these refactorings

commit	inputs	diffs
de8fdaacf	0	0
43ebb7	0	0
	0	0
	2	0
049bcd01d	trivial	0
90b52fde5	trivial	0
1c637de98	trivial	0
88805cbb3	0	0
4c4f50e1c	46	0

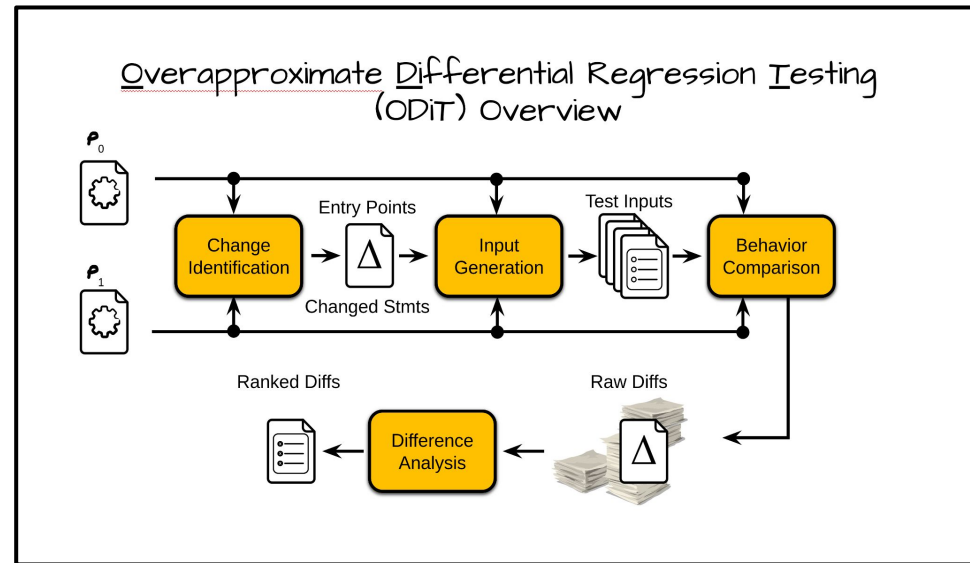
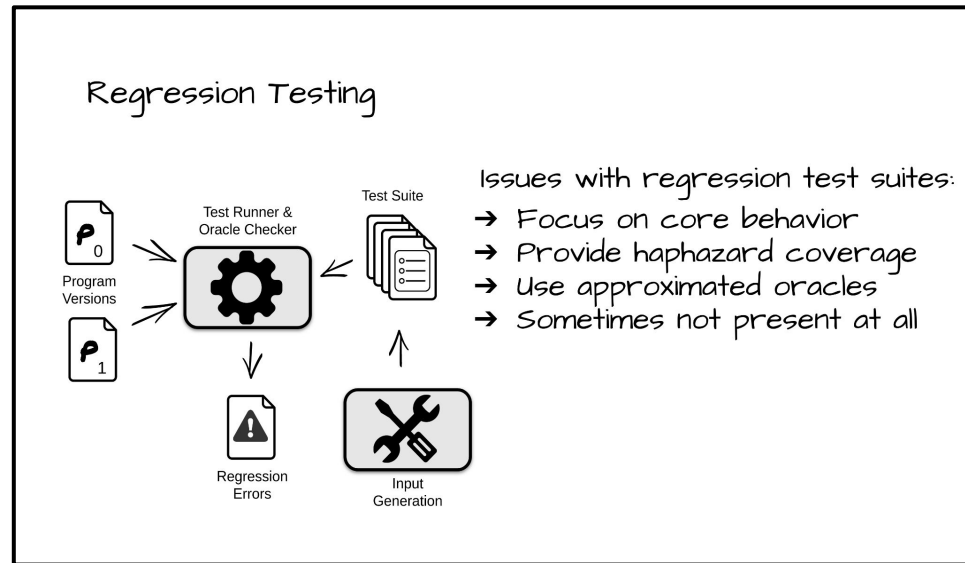
RQ3: Refactored Code

- 53 updates to Redis, from prior to 5.0.0 through 5.0.1
- Building commits that changed a program source file
- Manual inspection to categorize each commit

category	count	inputs	diffs
comment	8	N/A	0
data	5	unsupp	0
api	2	0	0
behavior	28	N/A	23
refactoring	10	7852	0

commit	inputs	diffs
de8fdaacf	906	0
43ebb7ee0	0	0
edc47a3ad	6898	0
3761582ff	trivial	0
50222af5f	2	0
b49bcd01d	trivial	0
90b52fde5	trivial	0
1c637de98	trivial	0
88805cbb3	0	0
4c4f50e1c	46	0

Conclusion



Evaluation: Implementation and Research Questions

Implementation:

- Program analysis and differencing: clang & llvm
- Symbolic execution engine: forked from KLEE 1.3

Research Questions:

- RQ1: Can ODIT detect and effectively rank regressions?
- RQ2: How does ODIT, which overapproximates behavior, compare to a tool that underapproximates behavior?
- RQ3: How does ODIT perform on refactored code?

RQ1: Detection and Ranking

benchmark	inputs	diffs	TP	FP	precision	rank
23-find	2552	67	1	66	1.5 %	67
24-find	22994	3	0	3	0.0 %	N/A
26-find	180975	65	10	55	15.4 %	1
27-find	7771	1	0	1	0.0 %	N/A
28-find	89420	4	0	4	0.0 %	N/A
30-find	35945					1
31-find						1
01-rm	671	0	0	0	-	N/A
03-cut	30641	5	0	5	0.0 %	N/A
04-tail	11407	1	1	0	100.0 %	1
05-tail	8311	1	0	1	0.0 %	N/A
06-cut	3198	5	2	3	40.0 %	1
07-seq	13427	2	1	1	50.0 %	1
08-seq	14088	3	1	2	33.3 %	1
09-seq	15248	2	2	0	100.0 %	1
10-cp	4239					1
12-cut						1
13-ls						1
14-ls						1
15-du						1
16-tail						1
17-cut						1
18-seq	15248	2	2	0	100.0 %	1
19-seq	8533	3	1	2	33.3 %	3
20-seq	15250	2	2	0	100.0 %	1
21-cut	18841	11	11	0	100.0 %	1
22-expr	2644	1	1	0	100.0 %	1
41-grep	27704	51	0	51	0.0 %	N/A
42-grep	2965	15	13	2	86.7 %	1
44-grep	586	0	0	0	-	N/A
45-grep	3142	1	0	1	0.0 %	N/A
46-grep	9069	5	3	2	60.0 %	2
47-grep	25758	22	0	22	0.0 %	N/A
48-grep	25918	16	0	16	0.0 %	N/A
49-grep	58	0	0	0	-	N/A
51-grep	168	13	0	13	0.0 %	N/A
52-grep	2012	3	3	0	100.0 %	1

Summary:

- In 58% of the cases, actual regressions in Top-3 (often Top-1)
- In 42% of the cases, FPs ranked higher than TPs
- In 33% of the cases, no false positives

Results

bench mark	regression detection						rank	cmp	
	inputs	diffs	+o	PPV	-o	FDR		bklee	shdw
01-rm	671	0	0	-	0	-	N/A	X	X
03-cut	30641	5	0	0.0%	5	100.0%	N/A	X	X
04-tail	11407	1	1	100.0%	0	0.0%	1	✓	X
05-tail	83								
06-cut	31								
07-seq	134								
08-seq	140								
09-seq	152								
10-cp	4239	0	0	-	0	-	N/A	X	✓
12-cut	28606	7	3	42.9%	4	57.1%	3	✓	✓ ²
13-ls	13062	3	2	66.7%	1	33.3%	1	✓	✓
14-ls	10186	10	10	100.0%	0	0.0%	1	✓	X
15-du	1402	10	8	80.0%	2	20.0%	1	✓	X
16-tail	8296	1	0	0.0%	1	100.0%	N/A	X	✓ ¹
17-cut	28573	7	3	42.9%	4	57.1%	3	✓	✓ ²
18-seq	15248	2	2	100.0%	0	0.0%	1	✓	X
19-seq	8533	3	1	33.3%	2	66.7%	3	✓	X
20-seq	15250	2	2	100.0%	0	0.0%	1	✓	X
21-cut	18841	11	11	100.0%	0	0.0%	1	✓	✓
22-expr	2644	1	1	100.0%	0	0.0%	1	✓	X

bench mark	inputs	diffs	regression detection			FDR	rank	bklee
			+o	PPV	-o			
23-find	2552	67	1	1.5%	66	98.5%	67	✓
24-find	22994	3	0	0.0%	3	100.0%	N/A	X
26-find	180975	65	10	15.4%	55	84.6%	1	✓
27-find	7771	1	0	0.0%	1	100.0%	N/A	X
28-find	89420	4	0	0.0%	4	100.0%	N/A	X
30-find	35945	7	7	100.0%	0	0.0%	1	✓
						34.4%	1	✓
						77.8%	1	✓
						77.8%	1	✓
						100.0%	N/A	X
						100.0%	N/A	X
37-find	89012	2	2	100.0%	0	0.0%	1	✓
38-grep	4583	8	5	62.5%	3	37.5%	2	✓
41-grep	27704	51	0	0.0%	51	100.0%	N/A	X
42-grep	2965	15	13	86.7%	2	13.3%	1	✓
44-grep	586	0	0	-	0	-	N/A	X
45-grep	3142	1	0	0.0%	1	100.0%	N/A	X
46-grep	9069	5	3	60.0%	2	40.0%	2	✓
47-grep	25758	22	0	0.0%	22	100.0%	N/A	X
48-grep	25918	16	0	0.0%	16	100.0%	N/A	X
49-grep	58	0	0	-	0	-	N/A	X
51-grep	168	13	0	0.0%	13	100.0%	N/A	X
52-grep	2012	3	3	100.0%	0	0.0%	1	✓

- Automatically identified >50% known regressions
 - Reported higher ranked FP in only 18/43 cases

Results

bench mark	regression detection							cmp		regression detection							bklee	
	inputs	diffs	+o	PPV	-o	FDR	rank	bklee	shdw	inputs	diffs	+o	PPV	-o	FDR	rank		
01-rm	671	0	0	-	0	-	N/A	X	X	23-find	2552	67	1	1.5%	66	98.5%	67	✓
03-cut	30641	5	0	0.0%	5	100.0%	N/A	X	X	24-find	22994	3	0	0.0%	3	100.0%	N/A	X
04-tail	11407	1	1	100.0%						26-find	180975	65	10	15.4%	55	84.6%	1	✓
05-tail	8311	1	0	0.0%						27-find	7771	1	0	0.0%	1	100.0%	N/A	X
06-cut	3198	5	2	40.0%						28-find	89420	4	0	0.0%	4	100.0%	N/A	X
07-seq	13427	2	1	50.0%										100.0%	0	0.0%	1	✓
08-seq	14088	3	1	33.3%										15.6%	54	84.4%	1	✓
09-seq	15248	2	2	100.0%	0	0.0%	1	X	X					22.2%	7	77.8%	1	✓
10-cp	4239	0	0	-	0	-	N/A	X	✓					22.2%	7	77.8%	1	✓
12-cut	28606	7	3	42.9%	4	57.1%	3	✓	✓ ²	36-find	68074	4	0	0.0%	4	100.0%	N/A	X
13-ls	13062	3	2	66.7%	1	33.3%	1	✓	✓	37-find	89012	2	2	100.0%	0	0.0%	1	✓
14-ls	10186	10	10	100.0%	0	0.0%	1	✓	X	38-grep	4583	8	5	62.5%	3	37.5%	2	✓
15-du	1402	10	8	80.0%	2	20.0%	1	✓	X	41-grep	27704	51	0	0.0%	51	100.0%	N/A	X
16-tail	8296	1	0	0.0%	1	100.0%	N/A	X	✓ ¹	42-grep	2965	15	13	86.7%	2	13.3%	1	✓
17-cut	28573	7	3	42.9%	4	57.1%	3	✓	✓ ²	44-grep	586	0	0	-	0	-	N/A	X
18-seq	15248	2	2	100.0%	0	0.0%	1	✓	X	45-grep	3142	1	0	0.0%	1	100.0%	N/A	X
19-seq	8533	3	1	33.3%	2	66.7%	3	✓	X	46-grep	9069	5	3	60.0%	2	40.0%	2	✓
20-seq	15250	2	2	100.0%	0	0.0%	1	✓	X	47-grep	25758	22	0	0.0%	22	100.0%	N/A	X
21-cut	18841	11	11	100.0%	0	0.0%	1	✓	✓	48-grep	25918	16	0	0.0%	16	100.0%	N/A	X
22-expr	2644	1	1	100.0%	0	0.0%	1	✓	X	49-grep	58	0	0	-	0	-	N/A	X
										51-grep	168	13	0	0.0%	13	100.0%	N/A	X
										52-grep	2012	3	3	100.0%	0	0.0%	1	✓

Outperformed state-of-the-art technique used as a baseline.