

GORDIAN

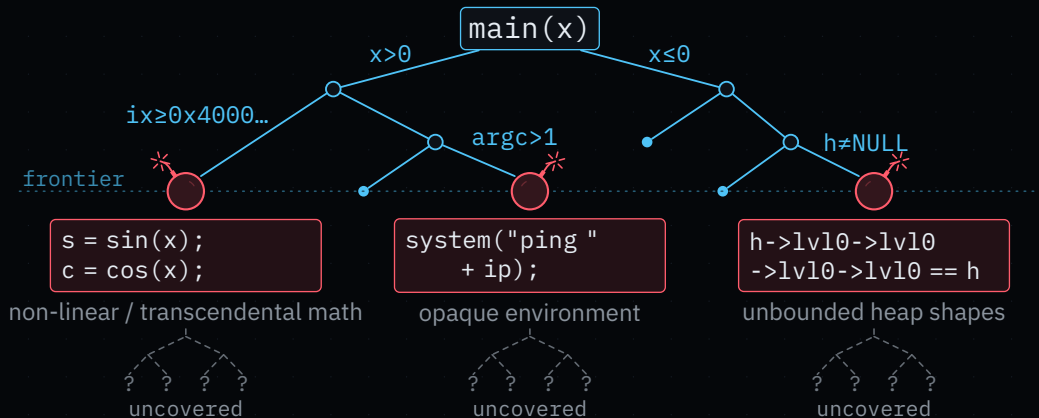
Defusing Logic Bombs in
Symbolic Execution with
LLM-Generated Ghost Code

Dimitrios Stamatios Bouras · Sergey Mechtaev
Peking University

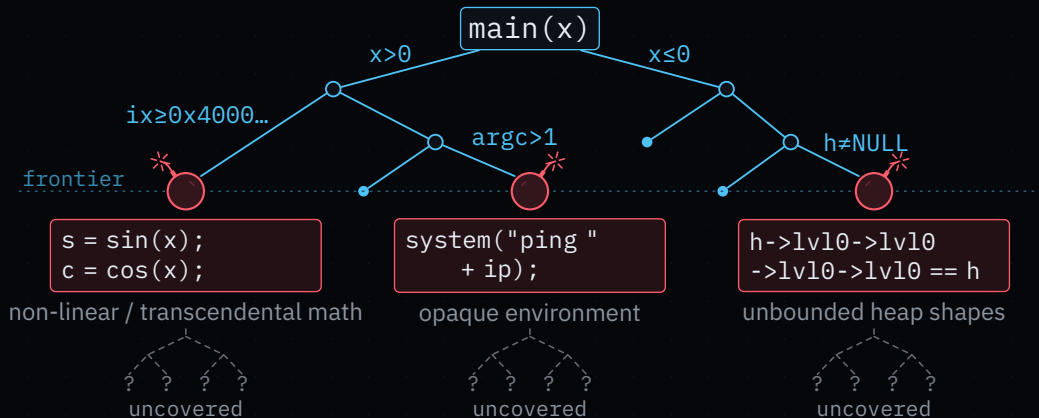
ISSTA 2026  KLEE Workshop 2026



└ Logic bombs stop symbolic execution



└ Logic bombs stop symbolic execution

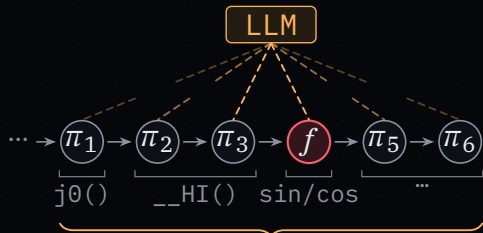


runs fine concretely — but yields **solver-hostile** path constraints

└ Replace the solver, or assist it?

LLM replaces solver

ConcoLLMic, PALM, AutoBug

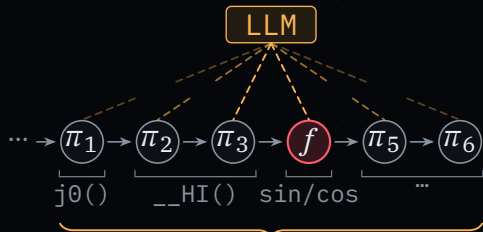


✗ global consistency lost
on deep paths

└ Replace the solver, or assist it?

LLM replaces solver

ConcoLLMic, PALM, AutoBug

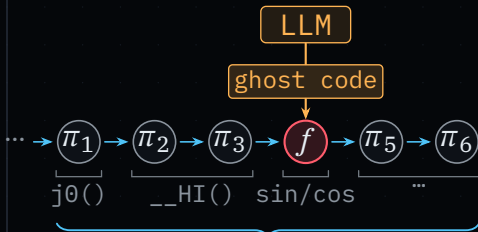


LLM: $\pi_1 \wedge \dots \wedge \pi_6$ at once

✗ global consistency lost
on deep paths

LLM assists solver

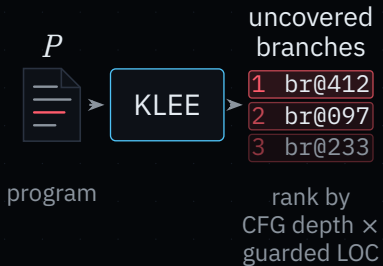
Gordian



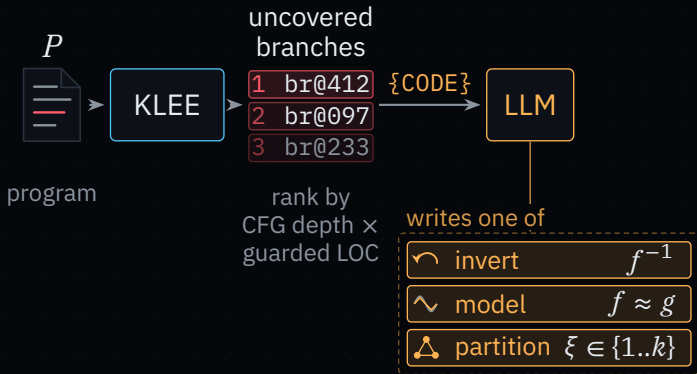
SMT: precise, global

✓ LLM local · solver global

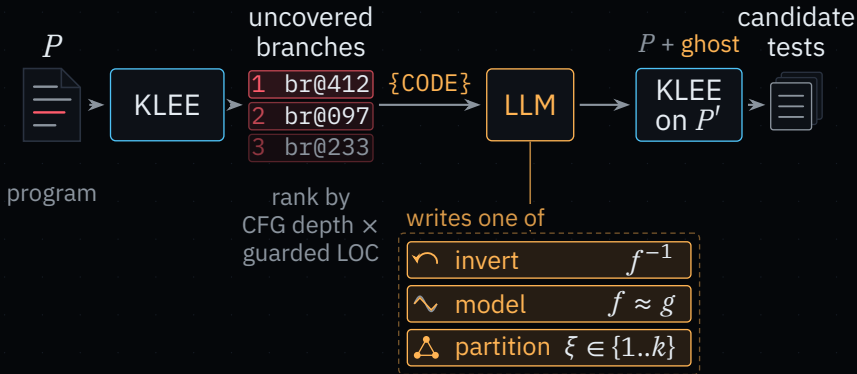
└ Ghost code as a solver aid



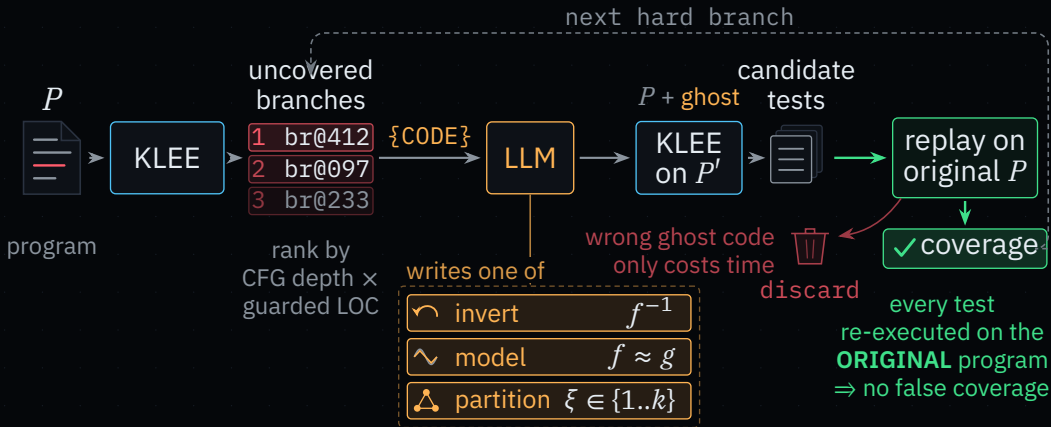
⊢ Ghost code as a solver aid



└ Ghost code as a solver aid



└ Ghost code as a solver aid



└ Example: __ieee754_j0 (FDLibM)

```
hx = __HI(x);  
ix = hx & 0x7fffffff;  
x = fabs(x);  
if (ix >= 0x40000000) {  
    s = sin(x);  
    c = cos(x);  
    ss = s - c;    cc = s + c;  
    if (ix < 0x7fe00000) {  
        z = -cos(x + x);  
        if ((s * c) < 0)  
            cc = z / ss;  
        else ss = z / cc;  
    }  
}
```

prefix

IEEE-754 bit masks: $|x| \geq 2$

hostile

sin, cos: no SMT theory

target

$(s * c) < 0$

✗ KLEE-Float

✗ ConcoLLMic

✓ Gordian

└ Cut out the hostile fragment

prefix

```
ix = __HI(x) & 0x7fffffff;  
if (ix >= 0x40000000)
```

hard fragment f

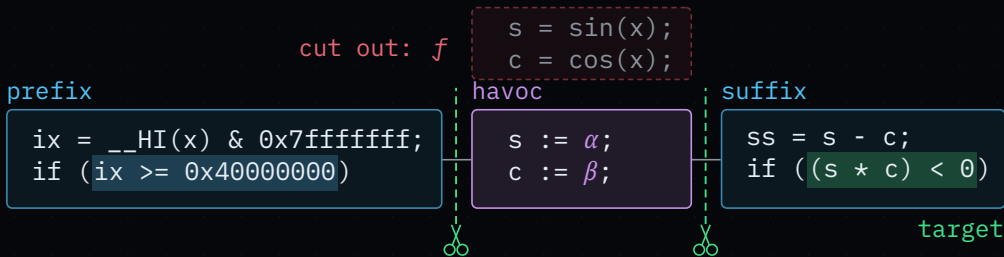
```
s = sin(x);  
c = cos(x);
```

suffix

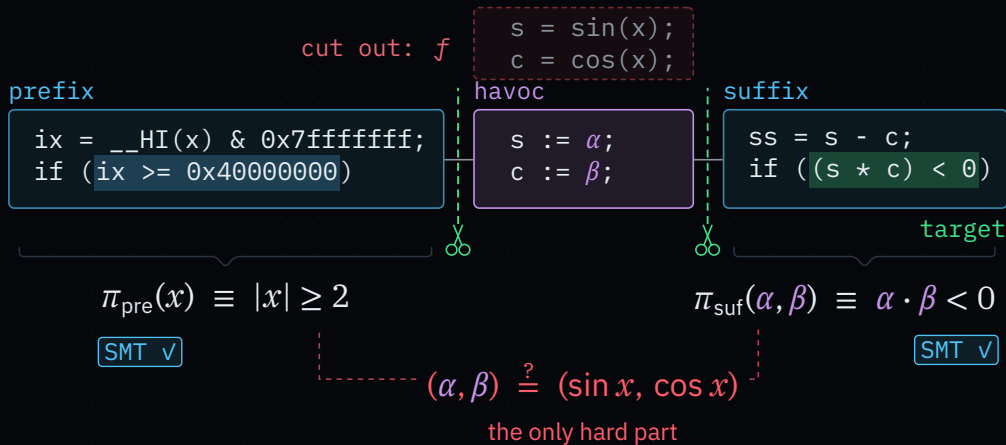
```
ss = s - c;  
if ((s * c) < 0)
```

target

└ Cut out the hostile fragment



└ Cut out the hostile fragment



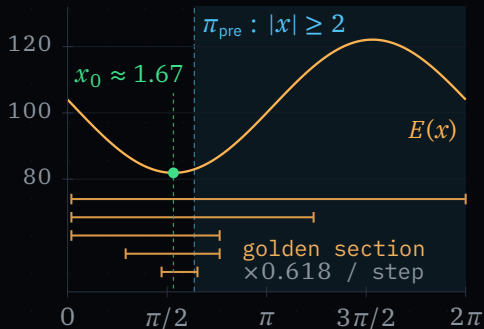
└ The LLM writes the inverse

ghost code | by the LLM

```
// |(sin x, cos x) - t|^2
double err(double x, double *t) {
    double ds = sin(x) - t[0];
    double dc = cos(x) - t[1];
    return ds*ds + dc*dc;
}
double f_inv(double s, double c) {
    double t[2] = {s, c};
    return golden(err, t, 0, 2*M_PI);
}
```

prompt: “invert the fragment; if exact inversion is impossible, implement a bounded numeric search”

$$f^{-1}(s=10, c=-1) = \arg \min_x E(x)$$



└ Bidirectional propagation

input space x



output space $(c, s) = f(x) = (\cos x, \sin x)$

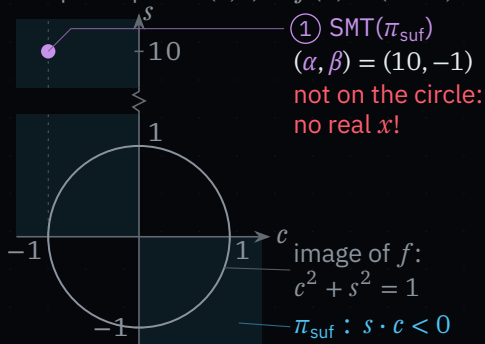


└ Bidirectional propagation

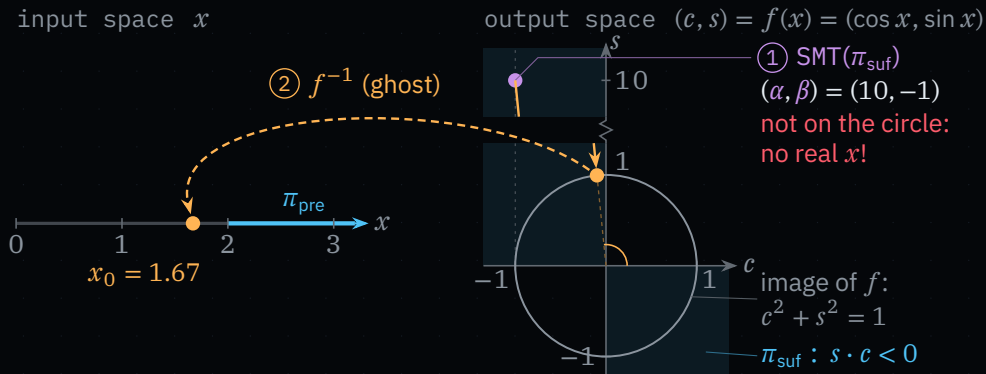
input space x



output space $(c, s) = f(x) = (\cos x, \sin x)$

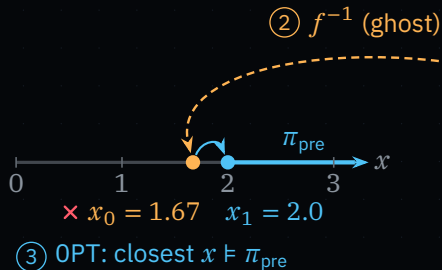


└ Bidirectional propagation

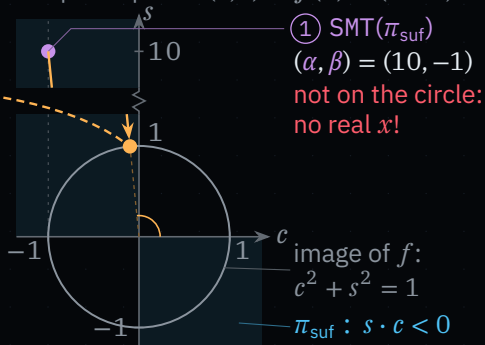


└ Bidirectional propagation

input space x



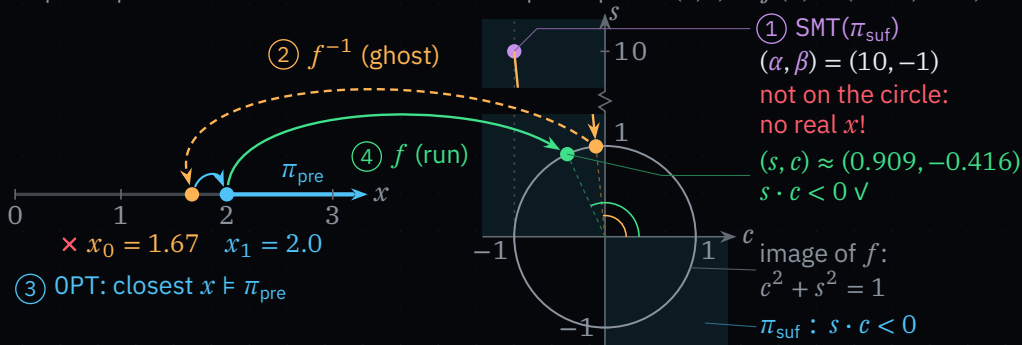
output space $(c, s) = f(x) = (\cos x, \sin x)$



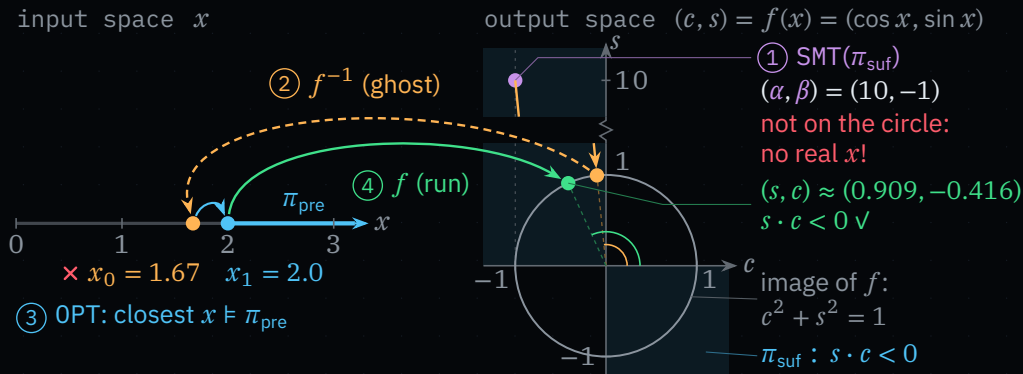
└ Bidirectional propagation

input space x

output space $(c, s) = f(x) = (\cos x, \sin x)$



└ Bidirectional propagation



⑤ replay $x=2.0$ on the original `__ieee754_j0` → target branch covered

KLEE-Float $\times$ ConcoLLMic $\times$ Gordian $\checkmark$

└ Surrogate models

original | opaque to KLEE

```
int ping(const char *ip) {  
    char cmd[128];  
    snprintf(cmd, sizeof(cmd),  
             "ping -c 1 %s", ip);  
    return system(cmd); // 0: up  
}  
  
int check_server(char *ip) {  
    if (ping(ip) == 0)  
        return 1;  
    return 0;  
}
```

ghost model | by the LLM

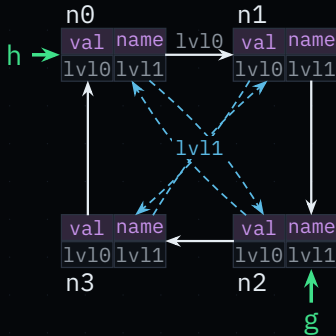
```
int ping(char *ip) {  
    if (!strcmp(ip, "127.0.0.1"))  
        return 0; // up  
    return 1;     // down  
}
```

- └ not equivalent —
branch-preserving
- └ solver-friendly theory
- └ "127.0.0.1" replayed
on the real ping ✓

└ Heap shapes explode

```
int proc_if_ring(SNode *h,  
                SNode *g) {  
    SNode *a = h ? h->lv10 : NULL;  
    /* ... b, c, d likewise */  
    if (!h||!a||!b||!c||!d) return 0;  
    if (d != h) return 0;  
    if (h->lv11!=b || a->lv11!=c ||  
        b->lv11!=d || c->lv11!=a)  
        return 0;  
    if (g->lv10->lv10!=h) return 0;  
    return proc_ring_values(h);  
}
```

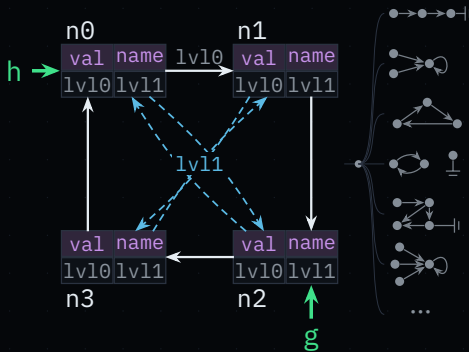
skiplist: lv10 next · lv11 express lane



⊢ Heap shapes explode

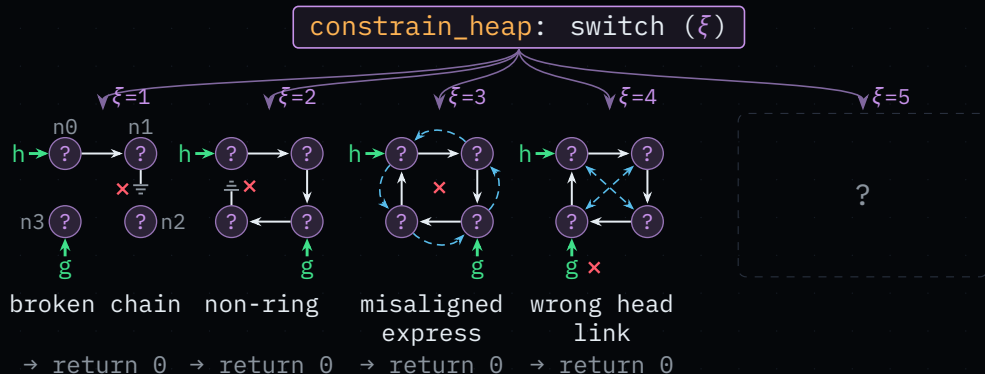
```
int proc_if_ring(SNode *h,
                 SNode *g) {
    SNode *a = h ? h->lv10 : NULL;
    /* ... b, c, d likewise */
    if (!h || !a || !b || !c || !d) return 0;
    if (d != h) return 0;
    if (h->lv11 != b || a->lv11 != c ||
        b->lv11 != d || c->lv11 != a)
        return 0;
    if (g->lv10->lv10 != h) return 0;
    return proc_ring_values(h);
}
```

skiplist: lv10 next · lv11 express lane



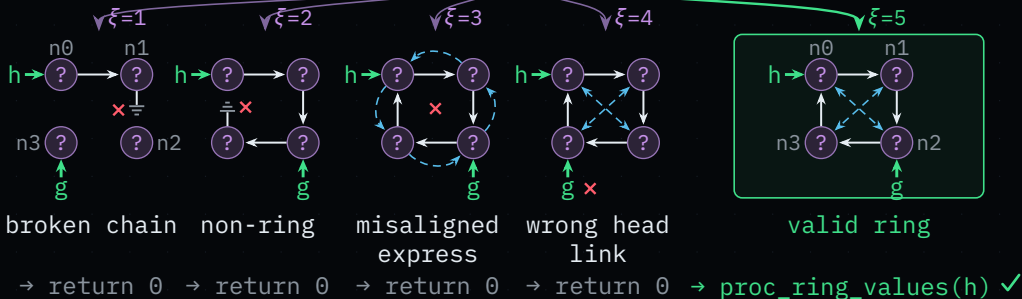
KLEE: 10 pointers → aliasing explosion → timeout (10 min)

└ Semantic heap topologies



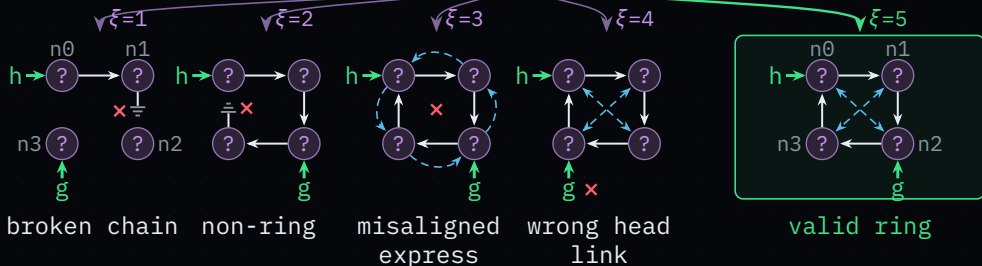
└ Semantic heap topologies

`constrain_heap: switch ( $\xi$ )`



└ Semantic heap topologies

constrain_heap: switch (ξ)



→ `return 0` → `return 0` → `return 0` → `return 0` → `proc_ring_values(h)` ✓

KLEE — $\times$ timeout (10 min) Gordian ✓ all branches in 2 s
links fixed · data symbolic — not tests, but families of executions

└ Evaluation

// benchmarks

LogicBombs

53 programs, one challenge
each

FDLibM

78 entry points, 5k LOC of float
math

Structured input

libexpat jq bc libyaml,
26k LOC

// baselines

KLEE 3.0

STP, Z3, KLEE-Float

ConcoLLMic

LLM-driven concolic

Cottontail

LLM + input grammars

// setup

GPT-5

Claude 3.7

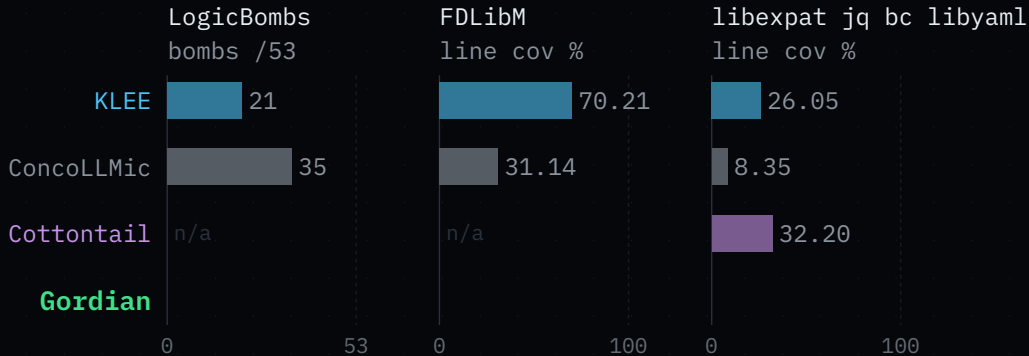
DeepSeek-Chat

Gordian = KLEE + Z3

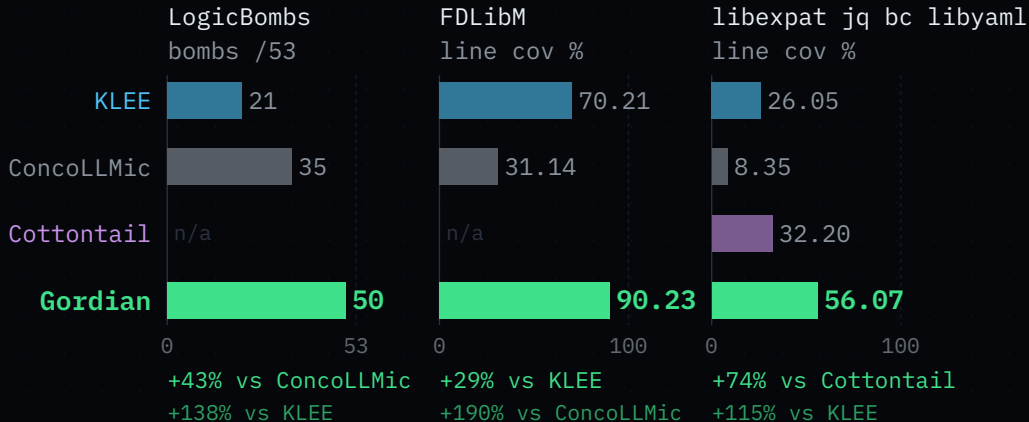
36 h per configuration

same KLEE options

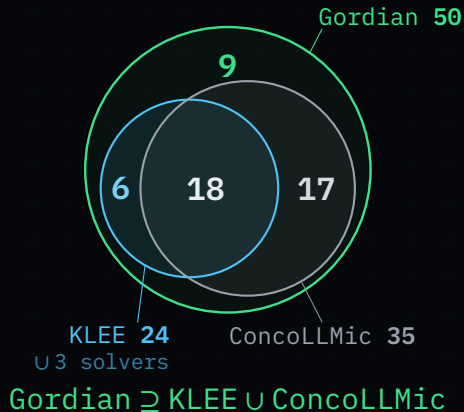
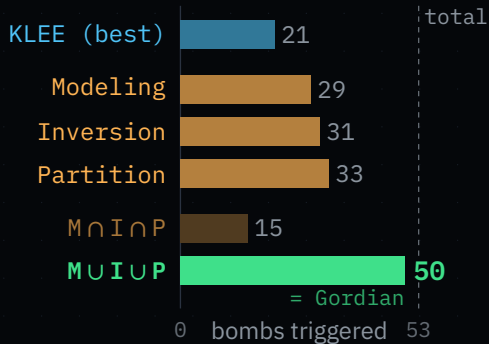
└ Coverage



└ Coverage

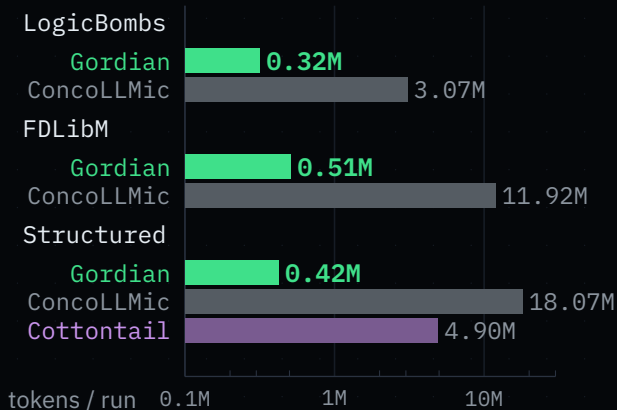


└ Complementary, strictly additive



each technique solves a different subset

└ 10× fewer tokens



26×

fewer than ConcoLLMic

12×

fewer than Cottontail

LLM ≈ 9% of runtime —
called at fragments, not paths

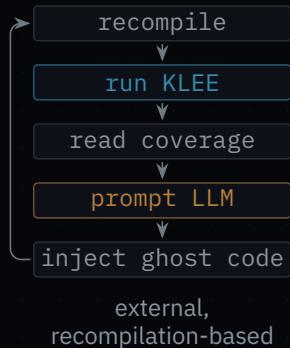
└ Toward ghost code inside KLEE

today / Gordian

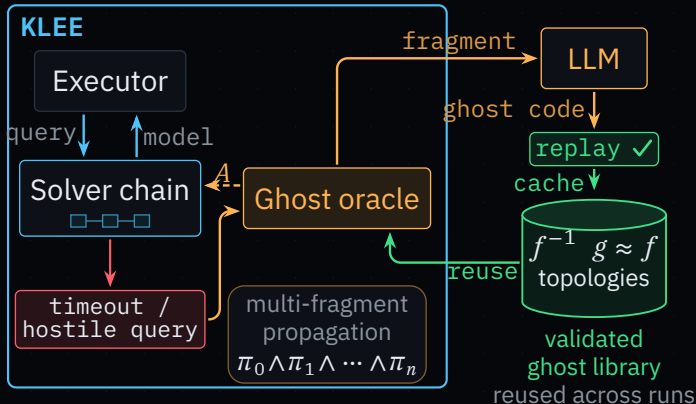


└ Toward ghost code inside KLEE

today / Gordian

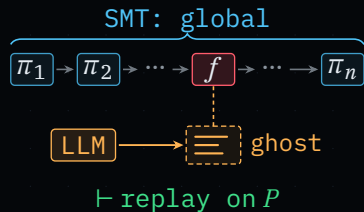


next / ghost code inside KLEE



└ Contributions

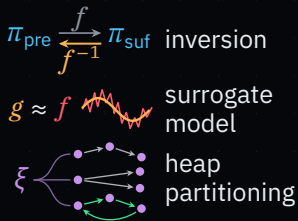
Ghost code for symbolic execution



LLM-generated solver aids at solver-hostile fragments; SMT reasoning remains global.

Sound: every input is validated on the original program.

Three ghost-code transformations



Bidirectional constraint propagation, solver-friendly surrogates, semantic heap topologies.

Evaluation on KLEE

LogicBombs **50/53**
bombs · KLEE 21 · ConcoLLMic 35

FDLibM **90.2%**
lines · KLEE 70.2%

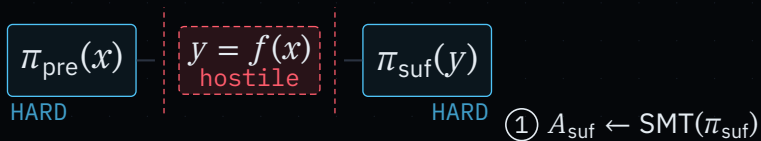
libexpat jq bc libyaml **56.1%**
KLEE 26.1% · Cottontail 32.2%

LLM tokens **-91-96%**
vs. LLM-based tools

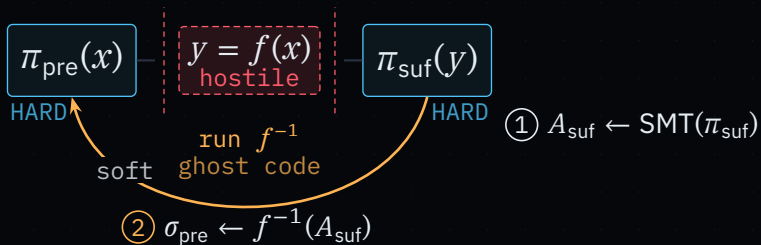
Coverage **+28.5-115.2%**
over KLEE, **+74.1-189.8%**
over LLM-based tools.

└ Questions?

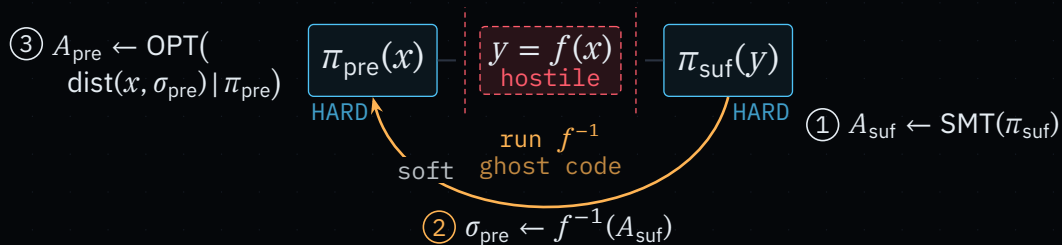
Backup: propagation algorithm



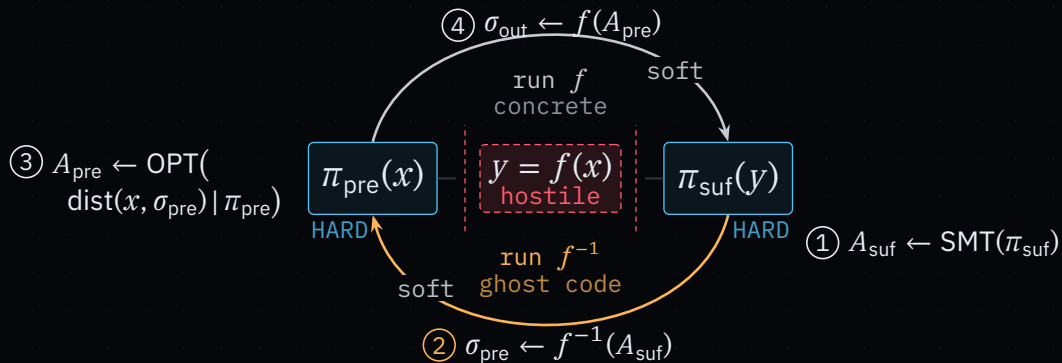
Backup: propagation algorithm



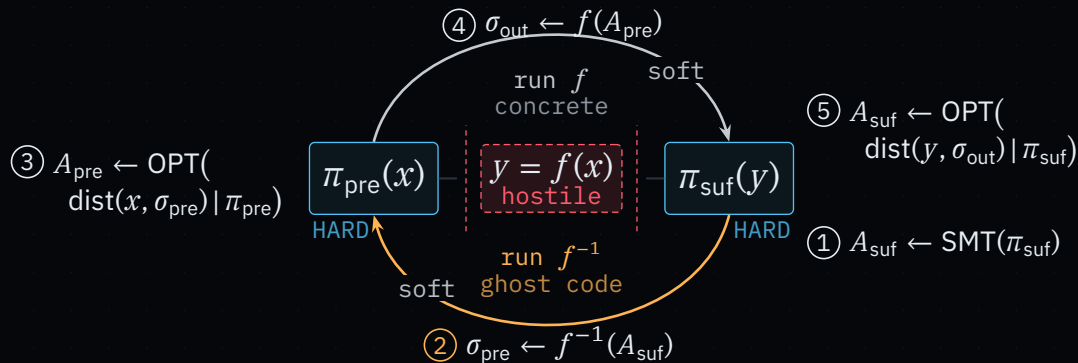
Backup: propagation algorithm



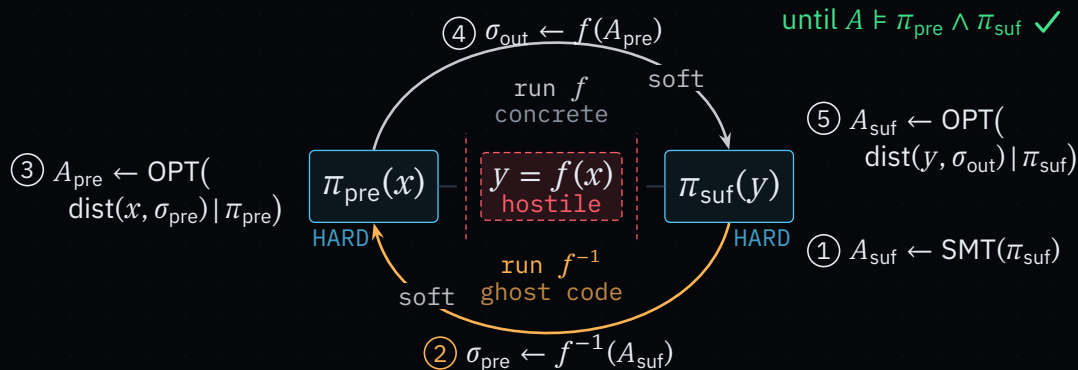
Backup: propagation algorithm



Backup: propagation algorithm



Backup: propagation algorithm



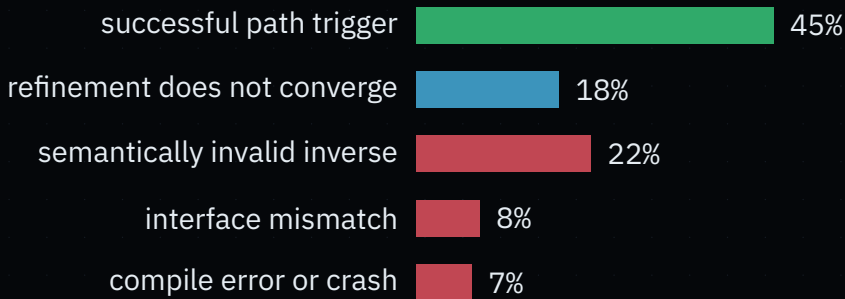
values crossing the cut: **soft targets**

$\pi_{\text{pre}}, \pi_{\text{suf}}$: **hard constraints**

one variable: $\min |A(x) - \sigma(x)|$ (ℓ_1 , Z3 Opt)

many variables: MaxSAT over soft equalities $x_i = \hat{x}_i$

└ Backup: LLM inversion quality



100 sampled inversion attempts (LogicBombs): 63% defect-free inverses, 45% end-to-end. Wrong inverses only cost time.

└ Backup: structured inputs

	KLEE-STP	ConcoLLMic	Cottontail	Gordian
libexpat	12.62	5.11	18.30	47.75
jq	29.80	10.00	31.74	45.37
bc	44.70	6.87	48.12	69.79
libyaml	23.74	29.17	55.24	65.59
overall	26.05	8.35	32.20	56.07

On libyaml, 19.2% of all covered lines are reached by Gordian alone.

└ Backup: limitations

- └ Recompilation-based orchestration around KLEE — build-system overhead.
- └ Sound, not complete: approximate inverses or surrogates may miss feasible paths.
- └ Propagation is heuristic optimisation — no convergence guarantee.
- └ One hostile fragment per path (multi-fragment extension: future work).
- └ Works best when hostility is localised to identifiable fragments.