

Stale pointers are the new black

Vincenzo Iozzo, Giovanni Gola

vincenzo.iozzo@zynamics.com

giovanni.gola@mail.polimi.it



Disclaimer

In this talk you won't see all those formulas, equations, code snippets and bullets.

From past experiences the speaker knows that all the aforementioned elements are no useful in helping people understand your idea.

You instead will see a few pictures which the speaker hopes will convey better the understanding of the ideas explained in the talk

You don't want slides like
this, do you?

Stale pointers are the new black

ZDI-09-041: Microsoft Internet Explorer 8 Rows Property Dangling Pointer Code Execution Vulnerability
<http://www.zerodayinitiative.com/advisories/ZDI-09-041>
June 10, 2009

-- CVE ID:
CVE-2009-1532

-- Affected Vendors:
Microsoft

-- Affected Products:
Microsoft Internet Explorer

Severity	CVE-2010-3257 High
Description	A code execution vulnerability exists in Apple Safari. The vulnerability is due when processing stale pointer issue with focusing. A remote attacker can entice a target user to open a maliciously crafted web page.

The specific flaw exists in js3250.dll. When a JavaScript object is created, there is not a proper sanity check for the object being passed to the JS_ValueToId() function. A remote attacker can exploit this vulnerability to execute arbitrary code on the SYSTEM user.

Title: TreeColumns dangling pointer
Impact: Critical
Announced: September 9, 2009
Reporter: TippingPoint ZDI
Products: Firefox

A vulnerability has been identified in Firefox. This issue is caused by a use-after-free error when processing certain JavaScript event objects, which could compromise a vulnerable system. This issue is caused by a use-after-free error in "mshtml.dll" when processing a specially crafted web page.

The flaw is caused due to a use-after-free error in WebKit when rendering HTML buttons, which could be exploited by attackers to execute arbitrary code via a specially crafted web page.

The specific flaw exists during deallocation of a CATtArray object. If the CATtArray object has been freed memory during the deallocation of the circular dereference. This can lead to code execution under the context of the currently logged in user.

Title: Dangling pointer crash regression from plugin parameter array fix
Impact: Critical
Announced: July 20, 2010
Reporter: Daniel Holbert
Products: Firefox 3.6.7

This vulnerability allows remote attackers to execute arbitrary code on vulnerable software utilizing Apple's WebKit library. User interaction is required to exploit this vulnerability in that the target must visit a malicious page.

The specific flaw exists in the handling of the run-in value for display CSS styles. A specially crafted web page can cause a use after free() condition in WebKit's WebCore::RenderBlock() method. This can be further leveraged by attackers to execute arbitrary code under the context of the current user.

II. DESCRIPTION

AVUPEN Vulnerability Research Team discovered a critical vulnerability in Microsoft Office Excel.

The vulnerability is caused by a dangling pointer when processing certain Formula records in an Excel file, which could be exploited by remote attackers to execute arbitrary code by tricking a user into opening a specially crafted Excel document.

Motivations



@0xcharlie
Charlie Miller

But srsly, how do you compete with [@taviso](#)? He reports all my Flash and Reader 0-days. Senseless slaughter of bugs, I'm quitting infosec.

17 Aug via web ☆ Favorite ↺ Retweet ↻ Reply

Different approaches then..



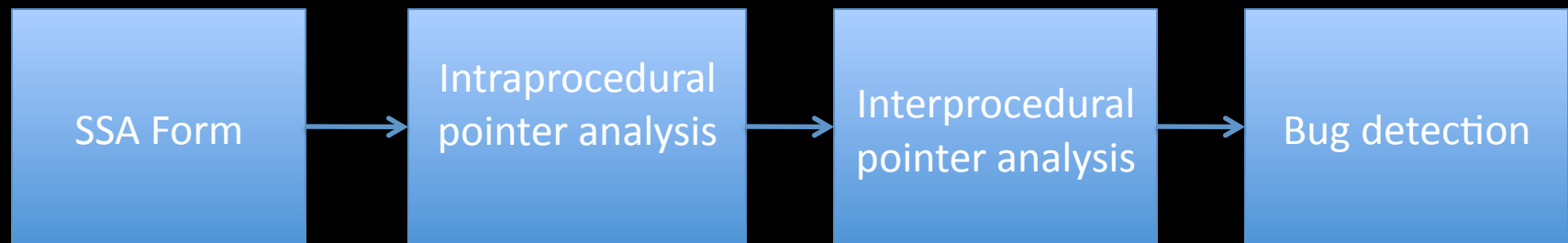
.. Or Static Analysis!

- Dataflow analysis
- Model Checking
- Theorem Proving

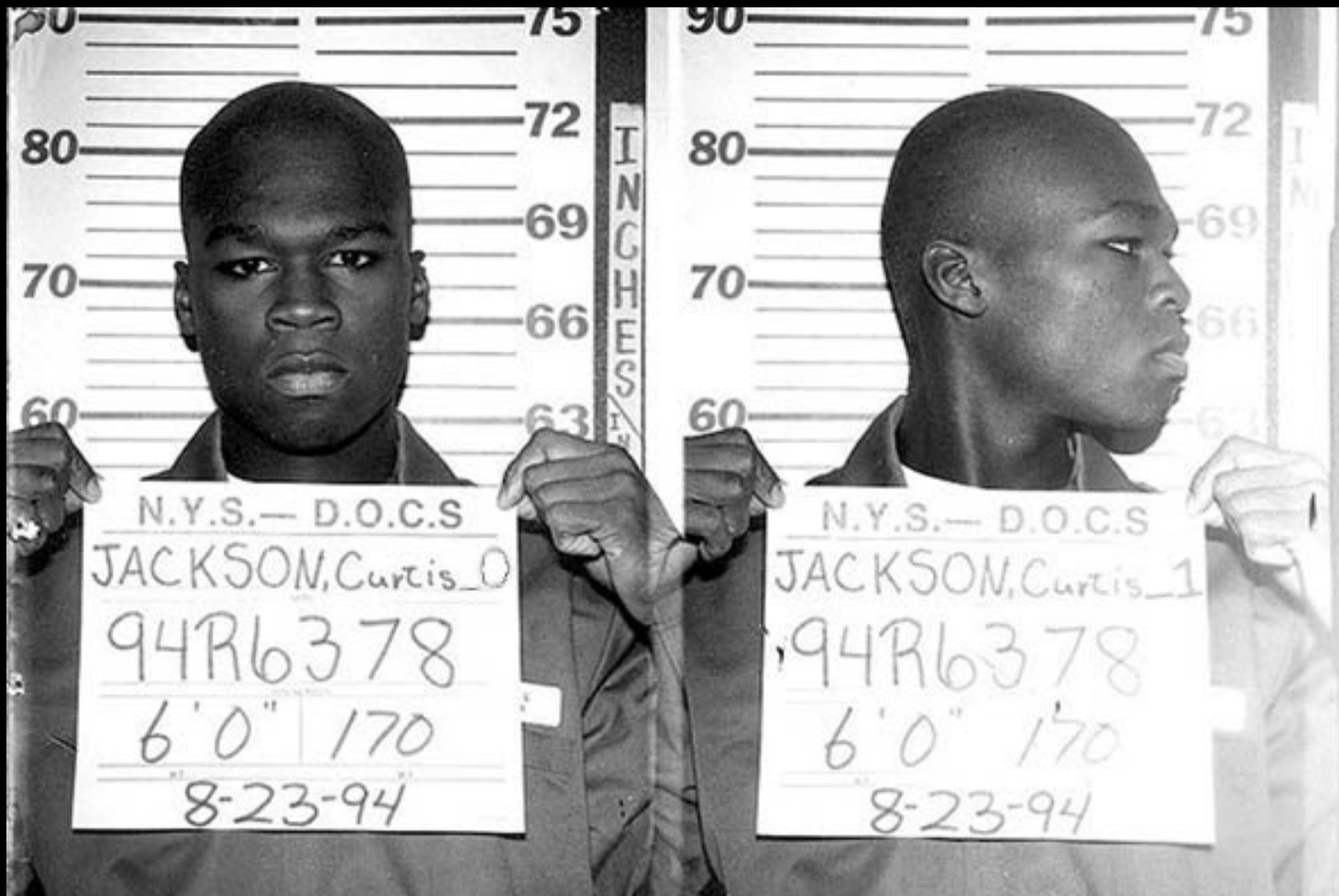
Our Idea



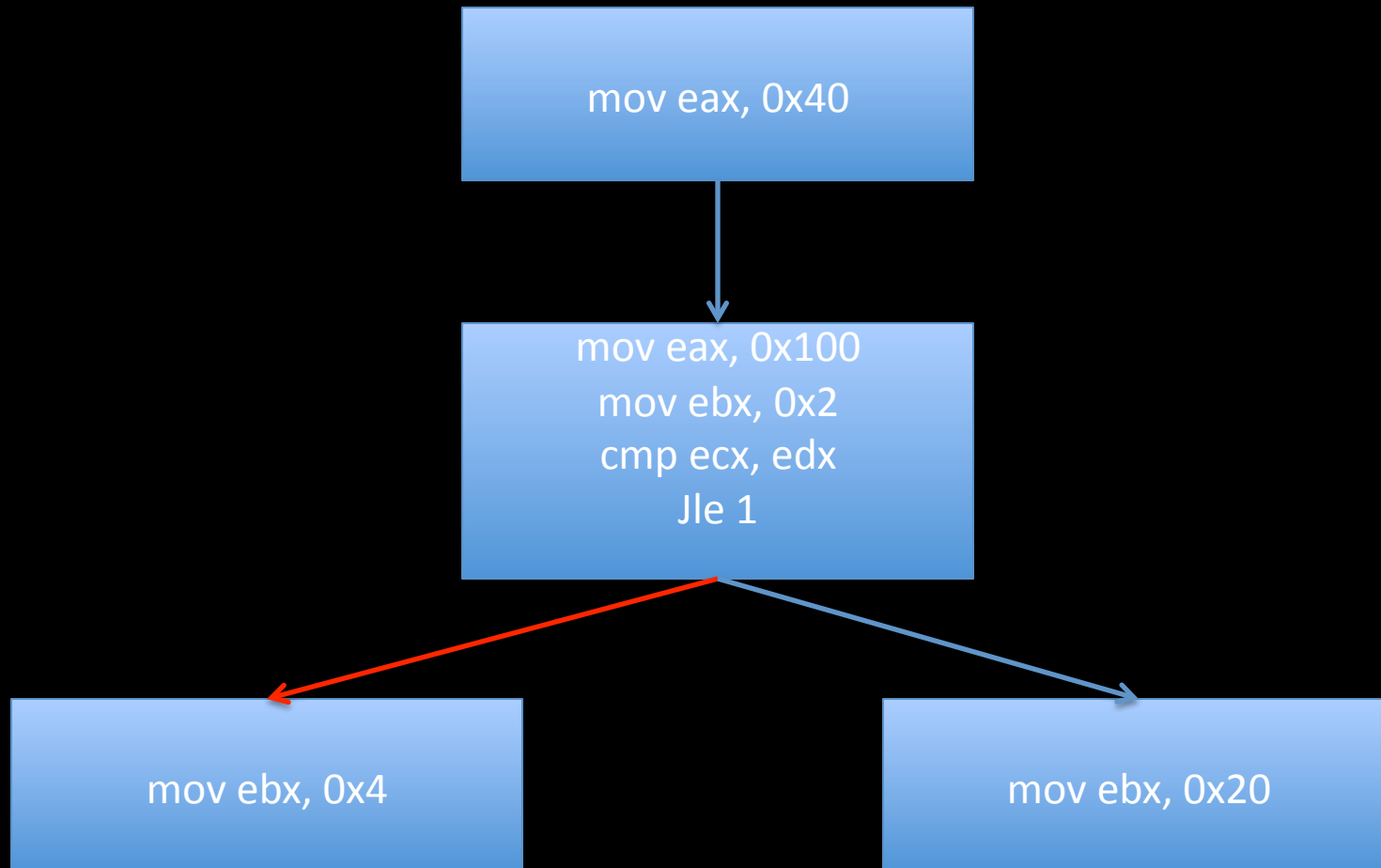
How it works



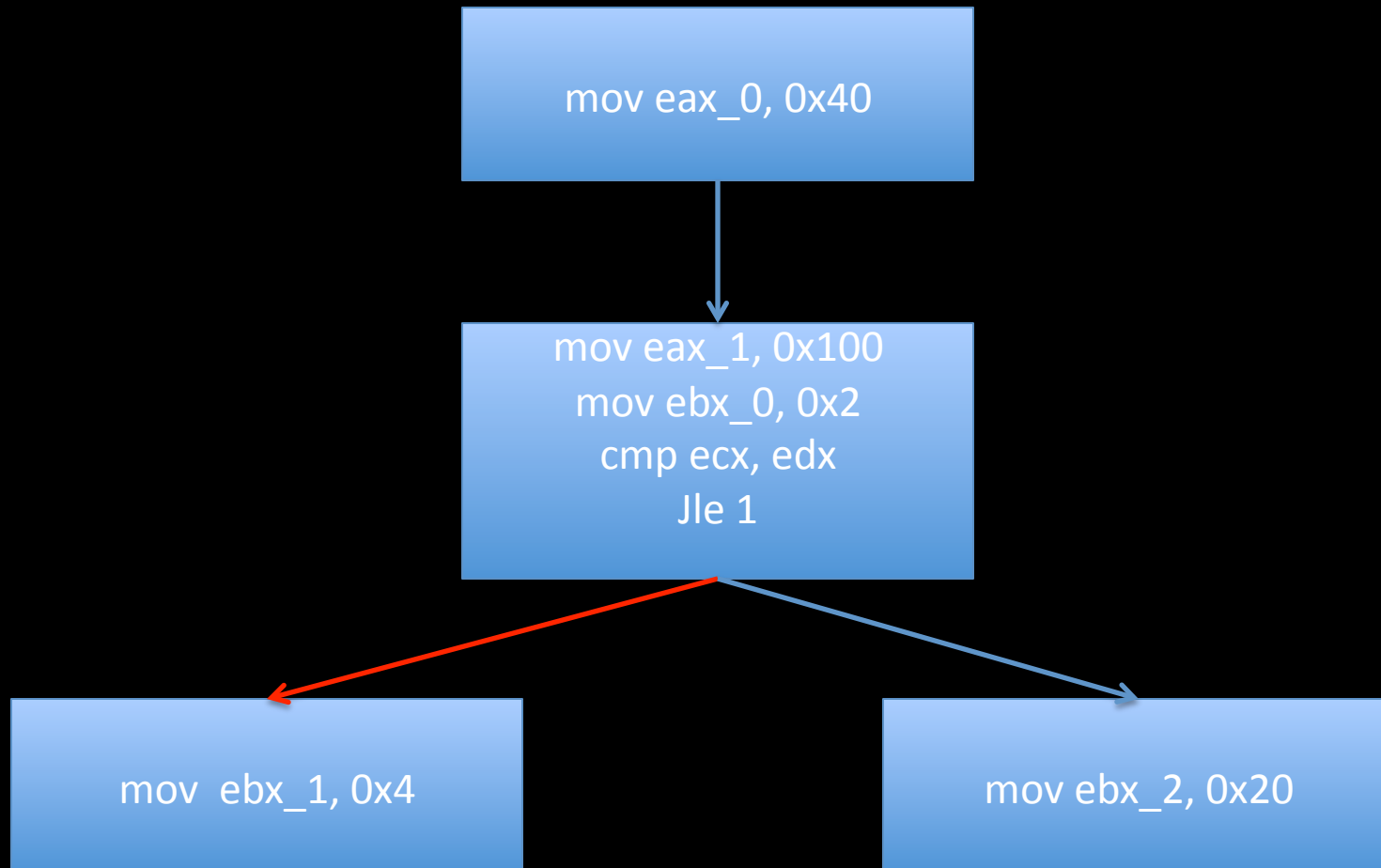
Single Static Assignment Form



Example

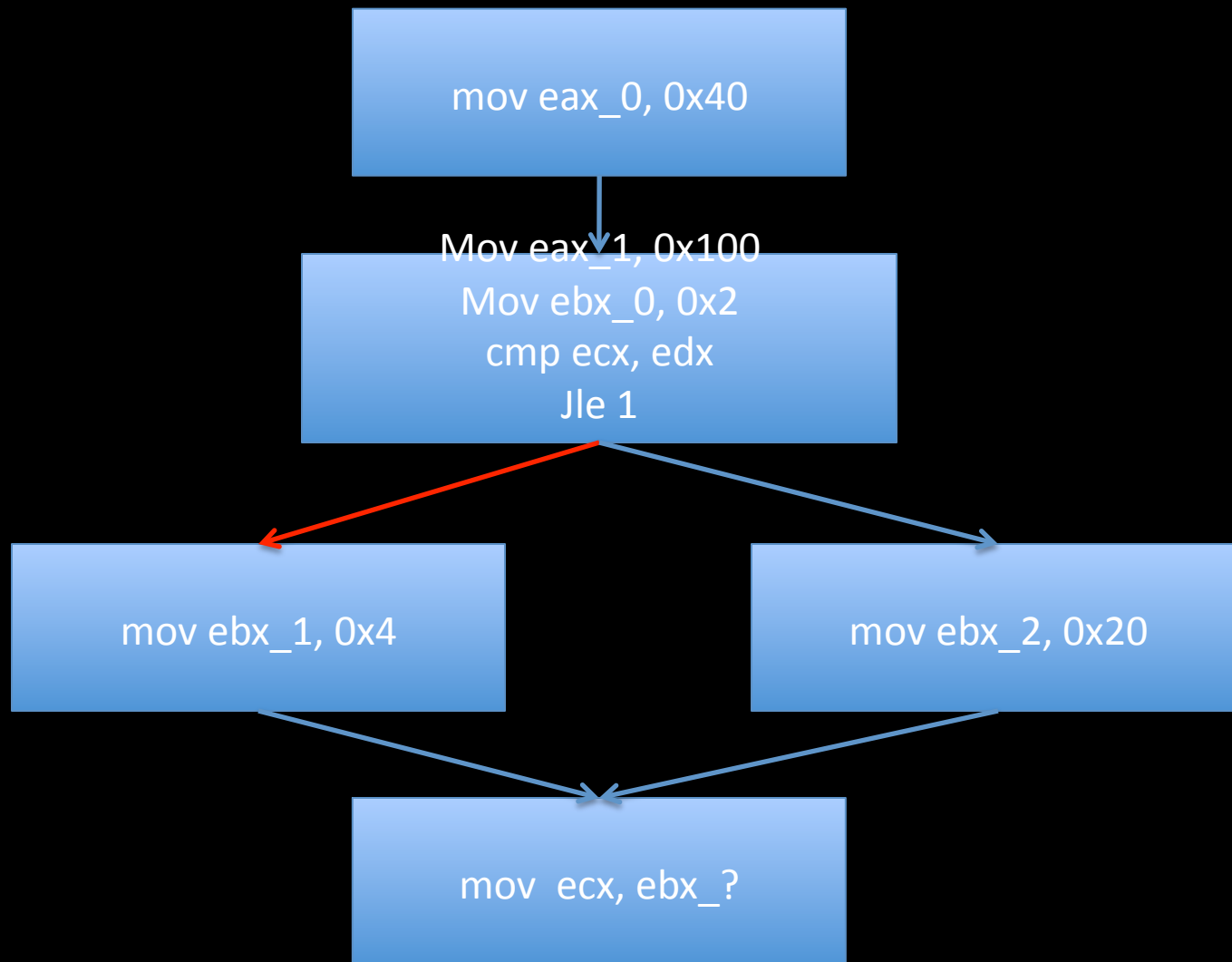


Looks better, right?

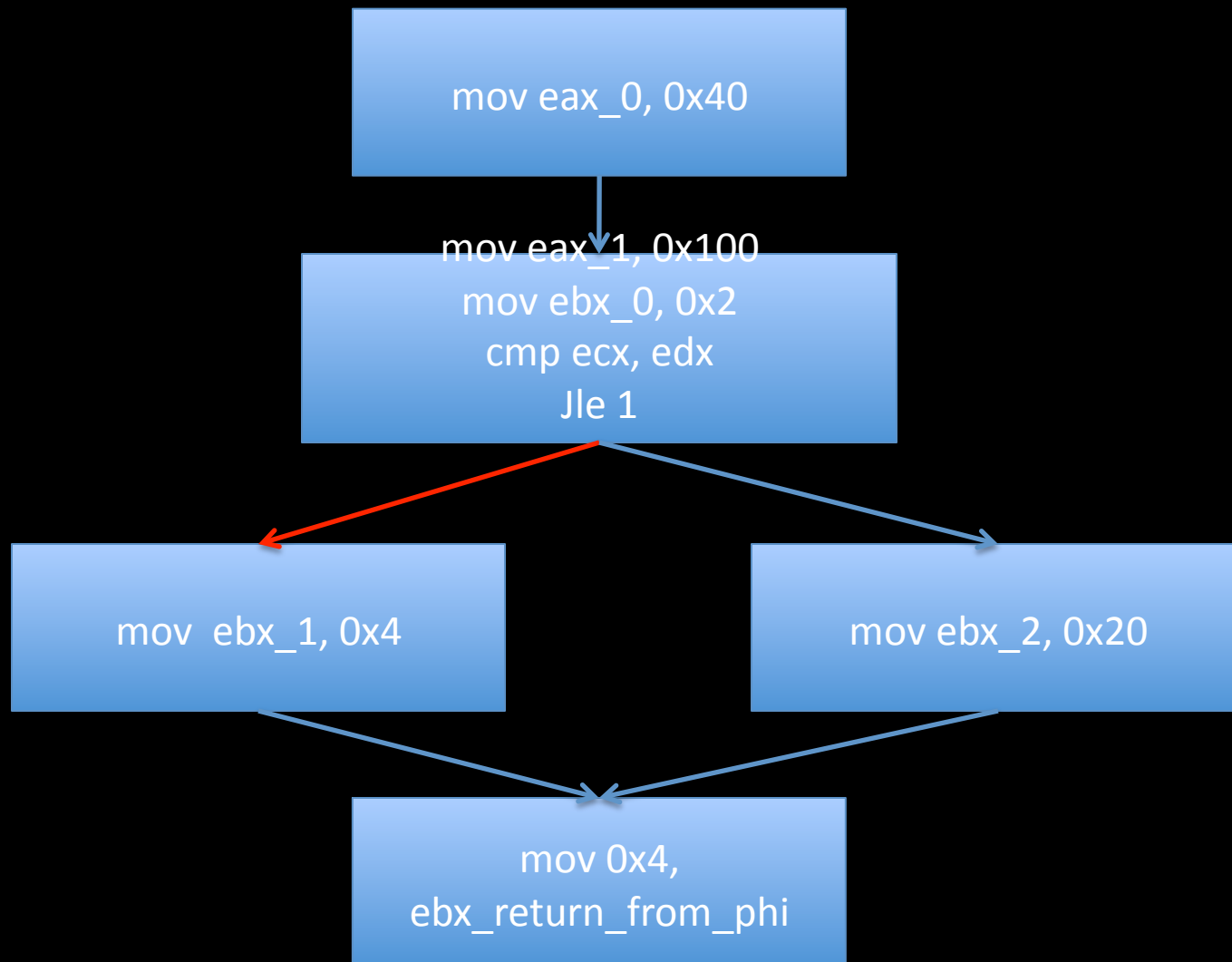




How about now?



Tah-dah!



Intermediate language interlude



Enter REIL

A small introduction to the REIL meta language

- small RISC instruction set (17 instructions)
 - Arithmetic instructions (ADD, SUB, MUL, DIV, MOD, BSH)
 - Bitwise instructions (AND, OR, XOR)
 - Logical instructions (BISZ, JCC)
 - Data transfer instructions (LDM, STM, STR)
 - Other instructions (NOP, UNDEF, UNKN)
- register machine
- unlimited number of temp registers
- side effect free
- no exceptions, floating point, 64Bit, ..

Example

```
00001EB0    WebCore.idb: :__ZNK7WebCore16AbstractDatabase14securityOriginEv
00001EB0    push     ebp
00001EB1    mov      ebp, esp
00001EB3    mov      eax, ss:[ebp+arg_0]
00001EB6    mov      eax, ds:[eax+0xC]
00001EB9    leave
00001EBA    retn
```


Translated

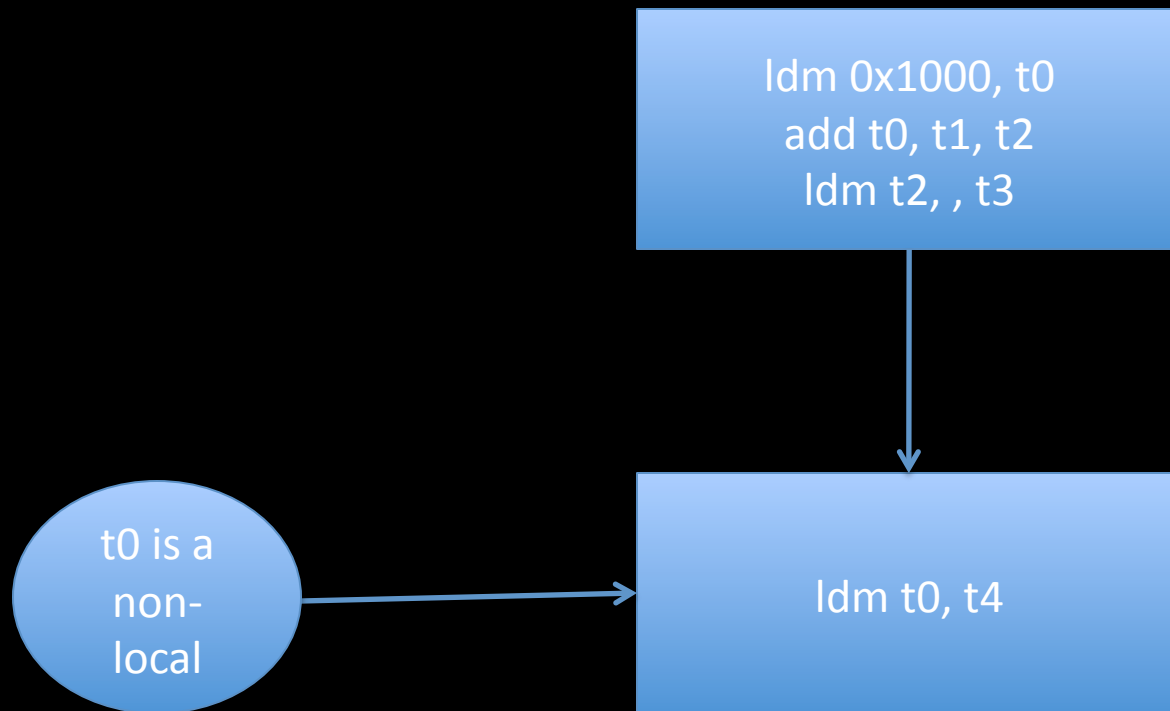
```
001EB000    sub        esp, 0x4, qword t0           // 00001EB0 push ebp
001EB001    and         qword t0, 0xFFFFFFFF, esp
001EB002    stm         ebp, , esp
001EB100    str         esp, , ebp           // 00001EB1 mov ebp, esp
001EB300    add         0x8, ebp, qword t0       // 00001EB3 mov eax, ss: [ebp + arg_0]
001EB301    and         qword t0, 0xFFFFFFFF, t1
001EB302    ldm         t1, , t2
001EB303    str         t2, , eax
001EB600    add         0xC, eax, qword t0       // 00001EB6 mov eax, ds: [eax + 12]
001EB601    and         qword t0, 0xFFFFFFFF, t1
001EB602    ldm         t1, , t2
001EB603    str         t2, , eax
001EB900    str         ebp, , esp           // 00001EB9 leave
001EB901    ldm         esp, , ebp
001EB902    add         esp, 0x4, qword t0
001EB903    and         qword t0, 0xFFFFFFFF, esp
001EBA00    ldm         esp, , t0           // 00001EBA retn
001EBA01    add         esp, 0x4, qword t1
001EBA02    and         qword t1, qword 0xFFFFFFFF, esp
001EBA03    jcc         0x1, , t0
```

Back to SSA

Flavours

- Non-pruned
- Semi-pruned
- Pruned

Non-locals



Algorithm

- Find non-locals
- Place phi-functions
- Recursively rename variables

A function

```
001E7000  sub     esp, 0x4, qword t0      // 00001E70 push ebp
001E7001  and     qword t0, 0xFFFFFFFF, esp
001E7002  stm     ebp, , esp
001E7100  str     esp, , ebp             // 00001E71 mov ebp, esp
001E7300  sub     esp, 0x4, qword t0      // 00001E73 call cs: 7800
001E7301  and     qword t0, 0xFFFFFFFF, esp
001E7302  stm     0x1E78, , esp
001E7303  jcc     0x1, , 0x1E78
```

```
001E7800  ldm     esp, , t0               // 00001E78 pop ecx
001E7801  add     esp, 0x4, qword t1
001E7802  and     qword t1, 0xFFFFFFFF, esp
001E7803  str     t0, , ecx
001E7900  add     0xD96D04, ecx, qword t0 // 00001E79 movzx eax, byte ds: [ecx + 14249220]
001E7901  and     qword t0, 0xFFFFFFFF, t1
001E7902  ldm     t1, , byte t2
001E7903  or      0x0, byte t2, eax
001E8000  str     ebp, , esp             // 00001E80 leave
001E8001  ldm     esp, , ebp
001E8002  add     esp, 0x4, qword t0
001E8003  and     qword t0, 0xFFFFFFFF, esp
001E8100  ldm     esp, , t0               // 00001E81 retn
001E8101  add     esp, 0x4, qword t1
001E8102  and     qword t1, qword 0xFFFFFFFF, esp
001E8103  jcc     0x1, , t0
```

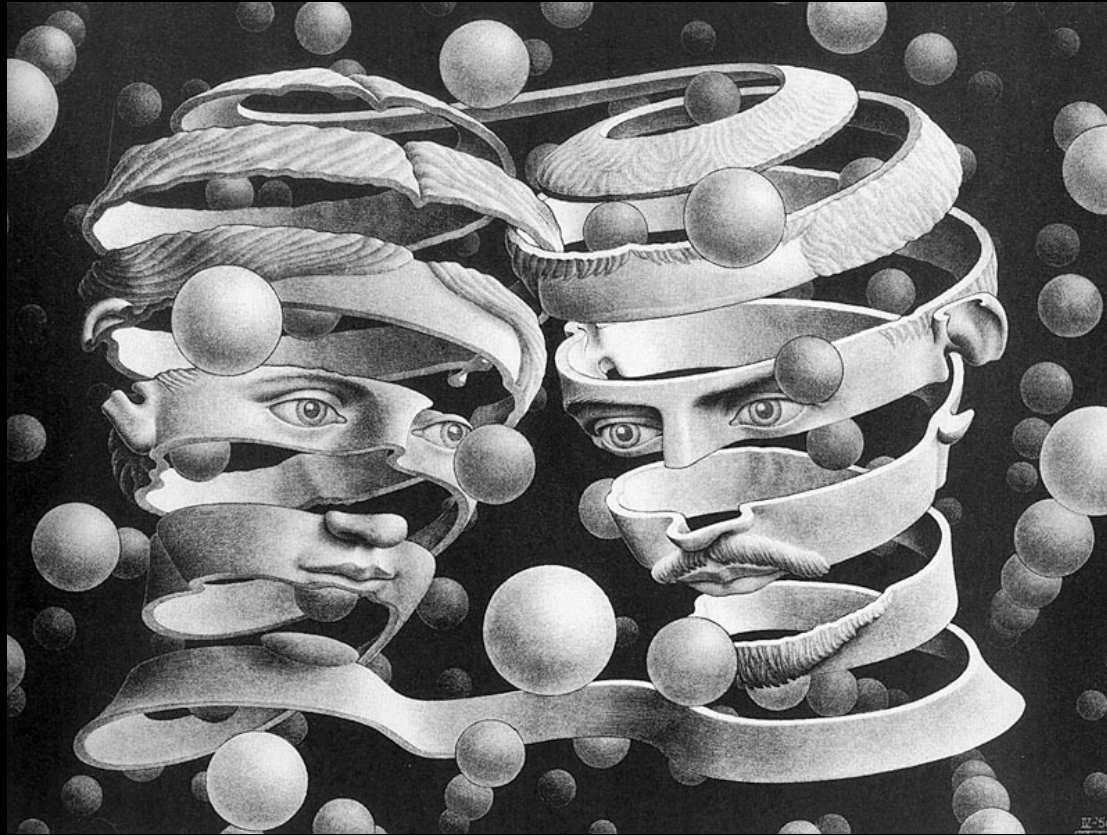
In SSA Form

```
001E7000    sub        esp_0, sub_4, qword t0_1
001E7001    and        qword t0_1, 0xFFFFFFFF, esp_1
001E7002    stm        ebp_0, , esp_2
001E7100    str        esp_2, , ebp_1
001E7300    sub        esp_2, sub_4, qword t0_2
001E7301    and        qword t0_2, 0xFFFFFFFF, esp_3
001E7302    stm        0x1E78, , esp_4
001E7303    jcc        0x1, , 0x1E78
```

```
001E7800    ldm        esp_4, , t0_3
001E7801    add        esp_4, sub_4, qword t1_1
001E7802    and        qword t1_1, 0xFFFFFFFF, esp_5
001E7803    str        t0_3, , ecx_1
001E7900    add        0xD96D04, ecx_1, qword t0_4
001E7901    and        qword t0_4, 0xFFFFFFFF, t1_2
001E7902    ldm        t1_2, , byte t2_1
001E7903    or         0x0, byte t2_1, eax_1
001E8000    str        ebp_1, , esp_6
001E8001    ldm        esp_6, , ebp_2
001E8002    add        esp_6, sub_4, qword t0_5
001E8003    and        qword t0_5, 0xFFFFFFFF, esp_7
001E8100    ldm        esp_7, , t0_6
001E8101    add        esp_7, sub_4, qword t1_3
001E8102    and        qword t1_3, qword 0xFFFFFFFF, esp_8
001E8103    jcc        0x1, , t0_6
```

Detour.. Abstract interpretation

Abstract Interpretation



Abstract Interpretation.. formally

Give several semantics linked by relations of abstraction

MonoREIL



monorail kitteh

experiencing sum technukul difukultiez

ICANHASCHEEZBURGER.COM 🐱💰🐱

So what you need?

- The control flow graph of a function
- A way to walk the CFG
- The lattice
 - Its elements
 - A way to combine lattice elements
- An initial state
- REIL instructions effects on the lattice

One constraint!

The lattice has to satisfy the ascending chain condition

Now the analysis itself

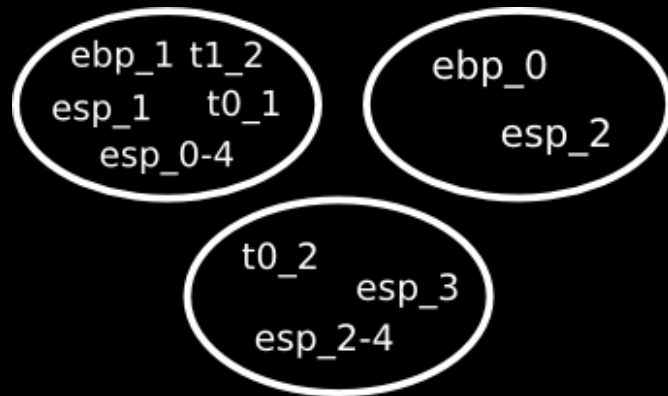


**CAUTION
HOT STUFF**

Intraprocedural Analysis

- Pointer Analysis: Efficiency
- Shape Analysis: Precision
- Alias Set Analysis: Tradeoff between the two

Data Structures



- push() and pop() on linked lists: 30% faster
- Hash consing: 30% memory saving

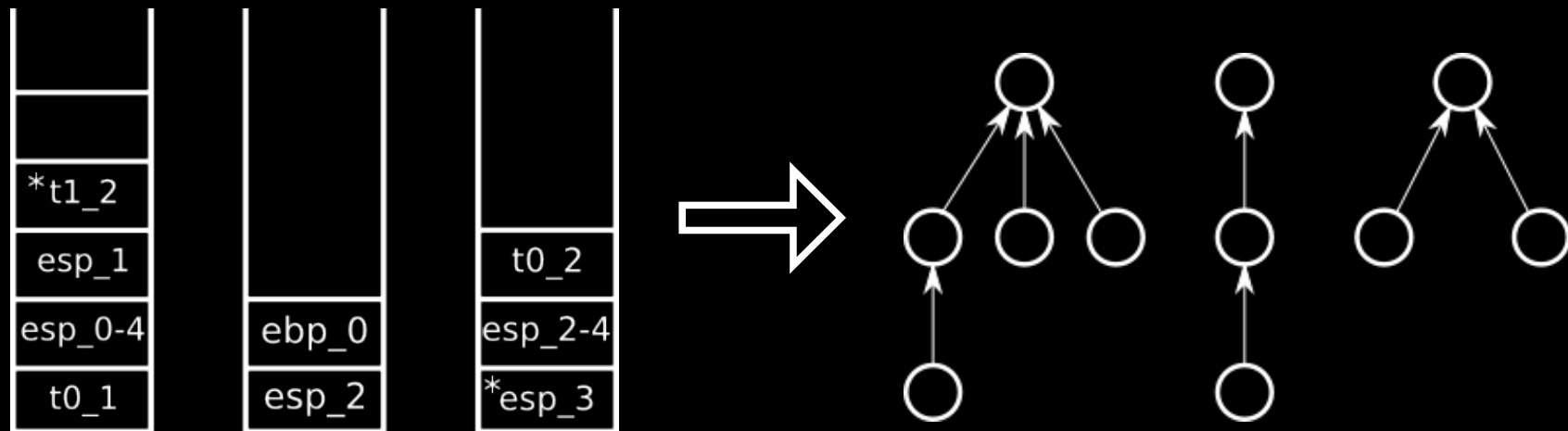
Transfer Functions

ARITHMETIC INSTRUCTIONS	OPERATION
ADD x_1, x_2, y	y is added to the alias set of $x_1 + x_2$
SUB x_1, x_2, y	y is added to the alias set of $x_1 - x_2$
MUL x_1, x_2, y	y is added to the alias set of $x_1 \cdot x_2$
DIV x_1, x_2, y	y is added to the alias set of $\left\lfloor \frac{x_1}{x_2} \right\rfloor$
MOD x_1, x_2, y	y is added to the alias set of $x_1 \bmod x_2$
BSH x_1, x_2, y	y is added to the alias set of $\begin{cases} x_1 \cdot 2^{x_2} & \text{if } x_2 \geq 0 \\ \left\lfloor \frac{x_1}{2^{-x_2}} \right\rfloor & \text{if } x_2 < 0 \end{cases}$
BITWISE INSTRUCTIONS	OPERATION
AND x_1, x_2, y	y is added to the alias set of $x_1 \& x_2$
OR x_1, x_2, y	y is added to the alias set of $x_1 x_2$
XOR x_1, x_2, y	y is added to the alias set of $x_1 \oplus x_2$
LOGICAL INSTRUCTIONS	OPERATION
BISZ x_1, ε, y	y is removed from all alias sets
JCC x_1, ε, y	does not affect alias sets
DATA TRANSFER INSTRUCTIONS	OPERATION
LDM x_1, ε, y	y is added to the alias set of $\text{mem}[x_1]$
STM x_1, ε, y	$\text{mem}[y]$ is added to the alias set of x_1
STR x_1, ε, y	y is added to the alias set of x_1
OTHER INSTRUCTIONS	OPERATION
NOP $\varepsilon, \varepsilon, \varepsilon$	does not affect alias sets
UNDEF $\varepsilon, \varepsilon, y$	y is removed from all alias sets
UNKN $\varepsilon, \varepsilon, \varepsilon$	does not affect alias sets

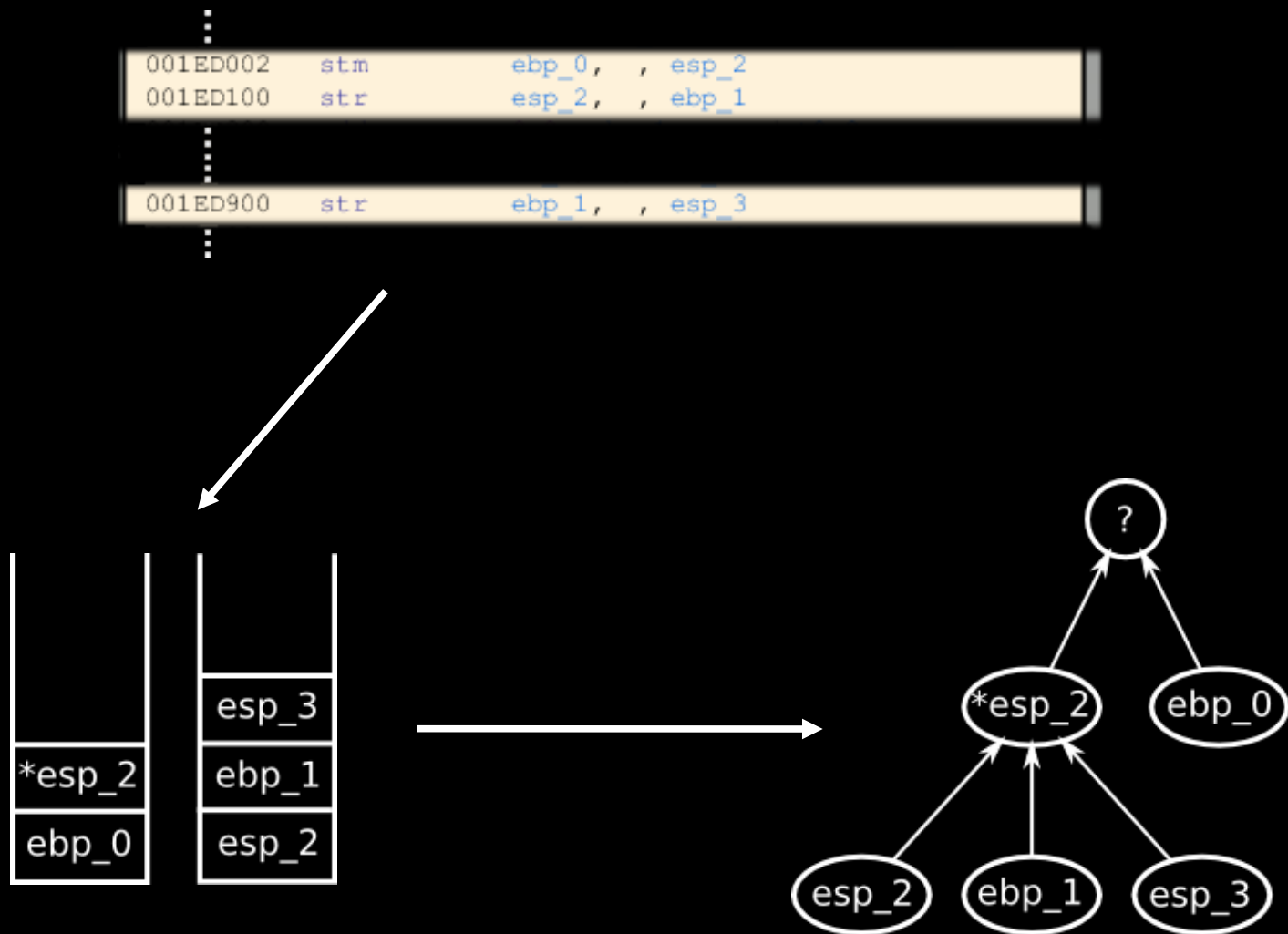
combine()

- Filter out non-live variables from each alias list:
 - $\text{live-out}(\text{inst}) \subseteq \text{vars}(\text{dom}(\text{inst}))$
 - Alias list $\cap \text{vars}(\text{sdom}(\Phi))$:
 - pop() from the list until
 $\text{top}(\text{alias list}) \in \text{vars}(\text{sdom}(\Phi))$
- Add aliases defined by Φ functions
- Unite the sets of lists

Data Structures Again



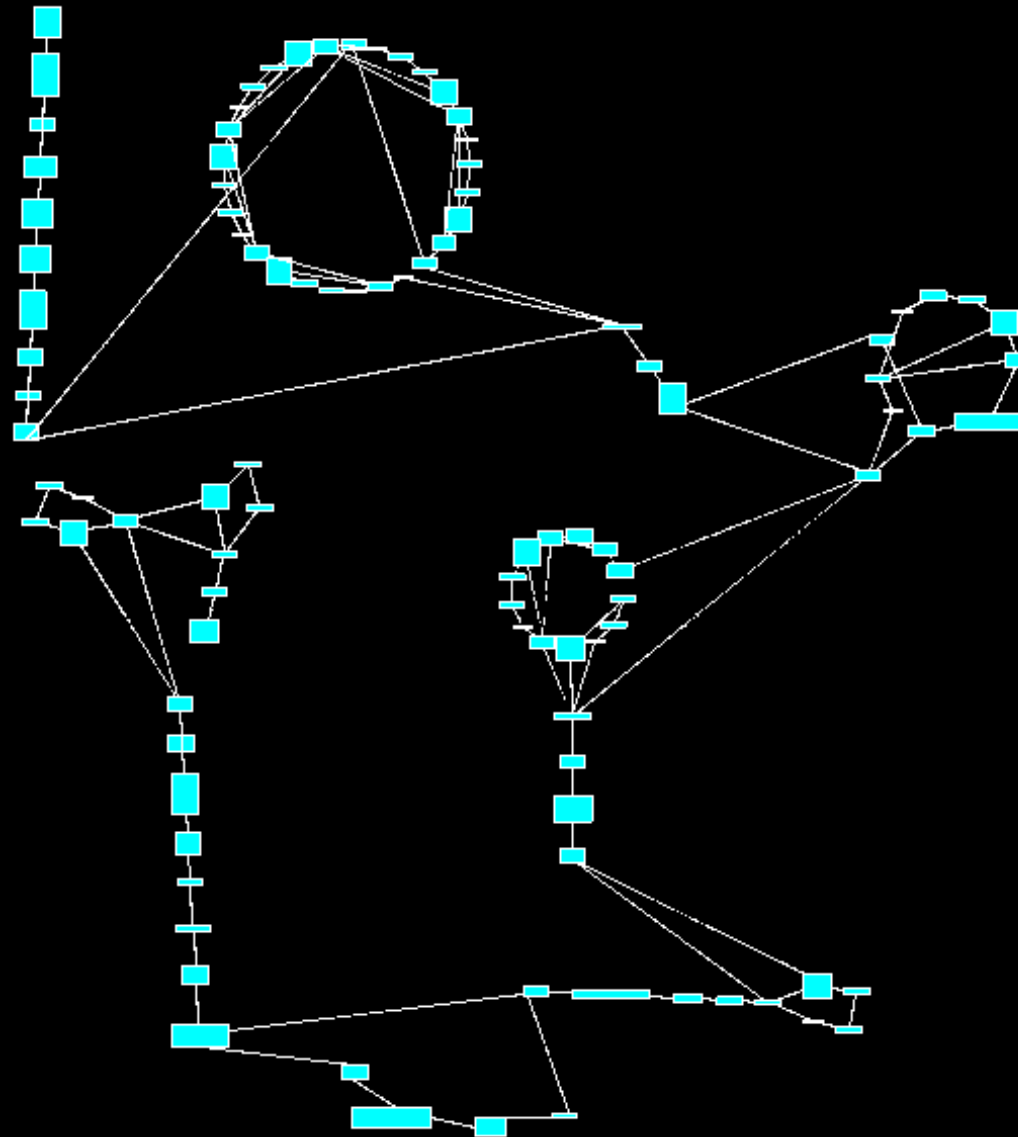
Example



Tracking parameters and return

- IDA effectively tracks parameters
- return is identified by guessing the calling convention

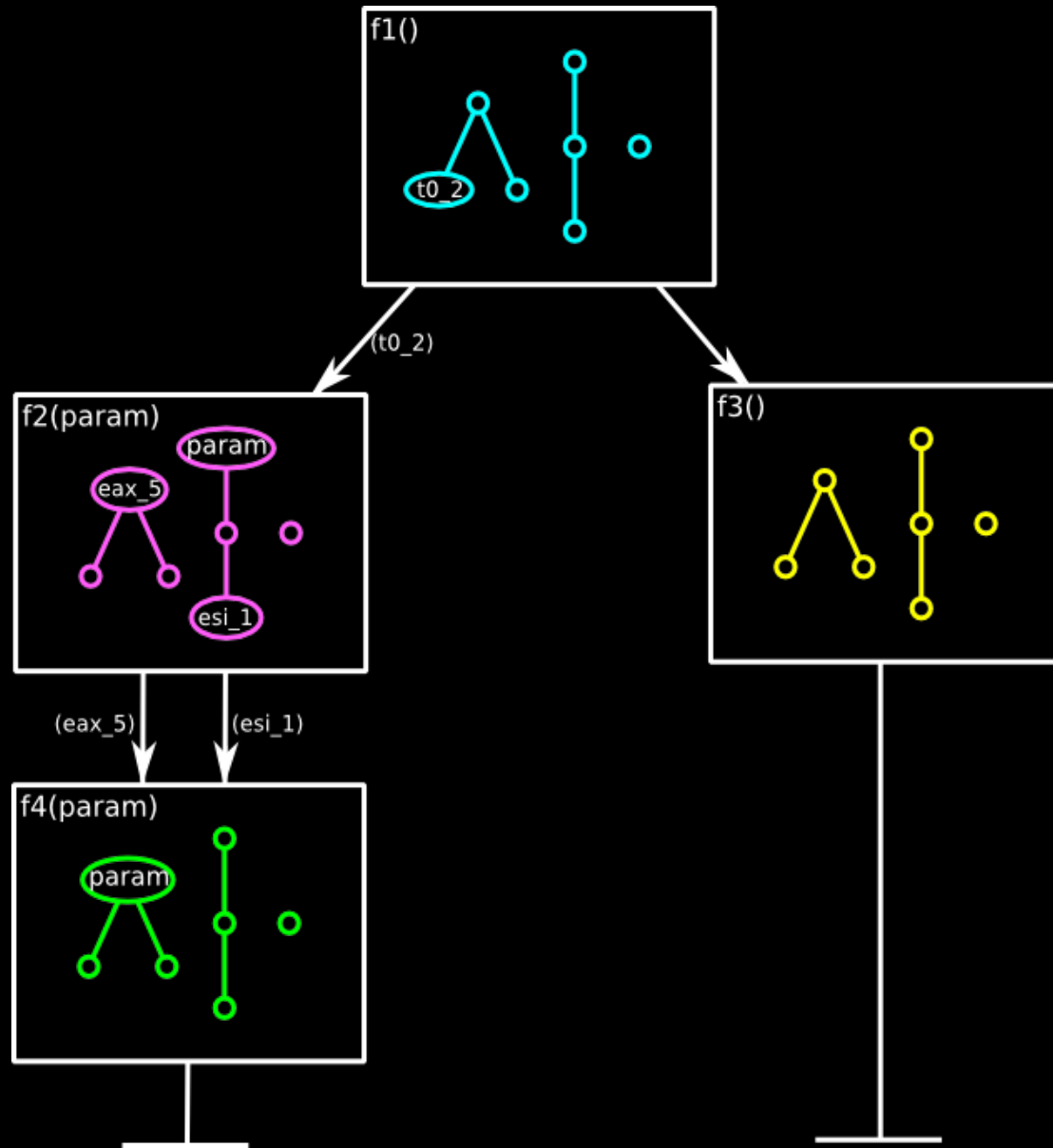
Interprocedural Analysis



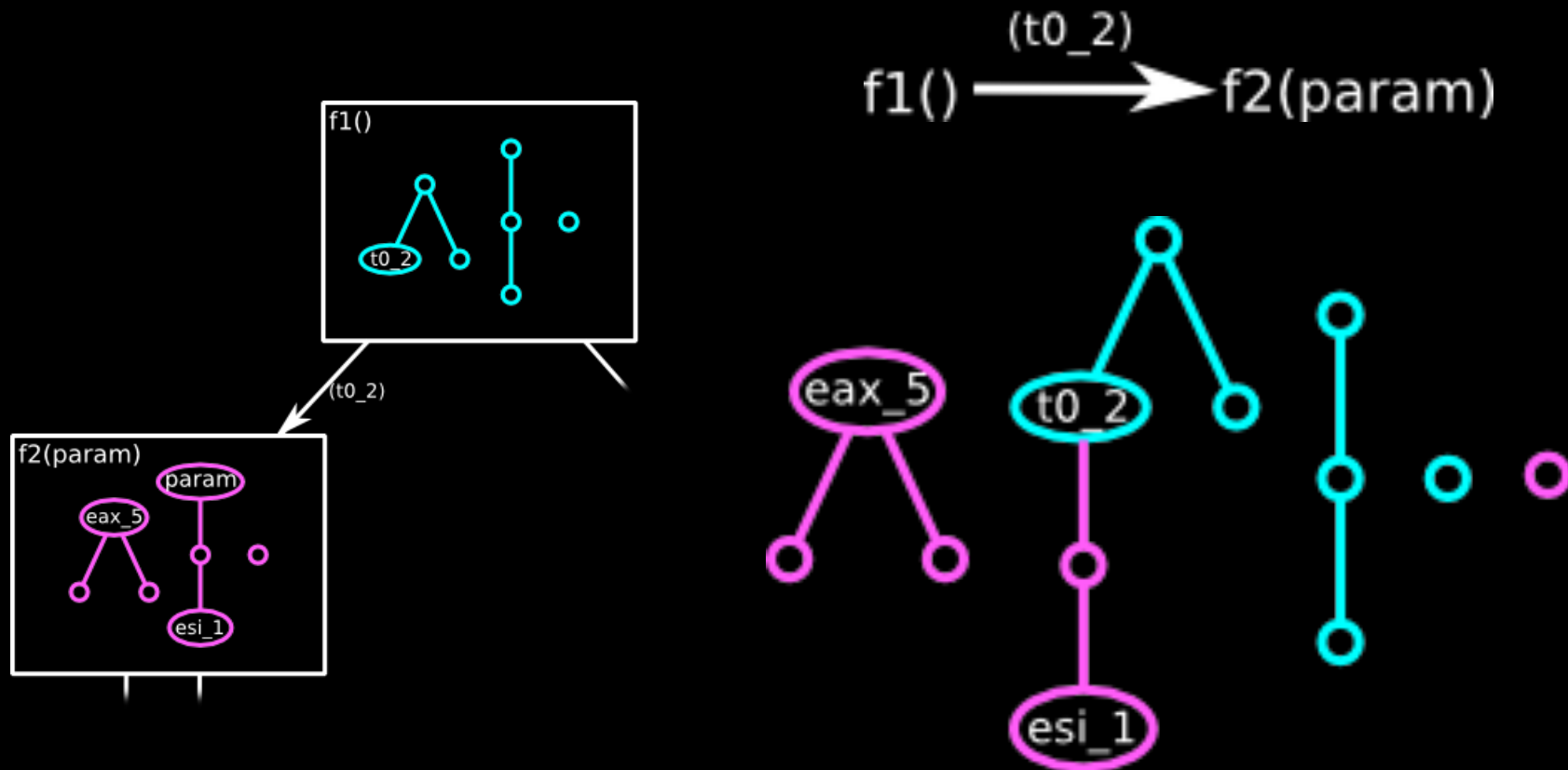
Algorithm

- Flow-insensitive
- Context-sensitive
- Implemented in BinNavi:
 - walks on the PCG

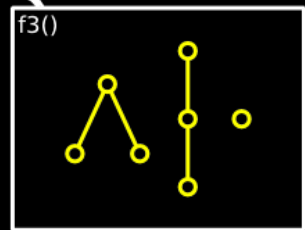
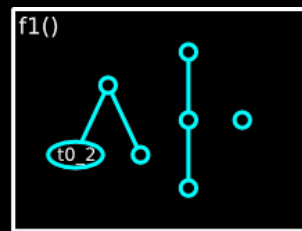
Procedure Call Graph



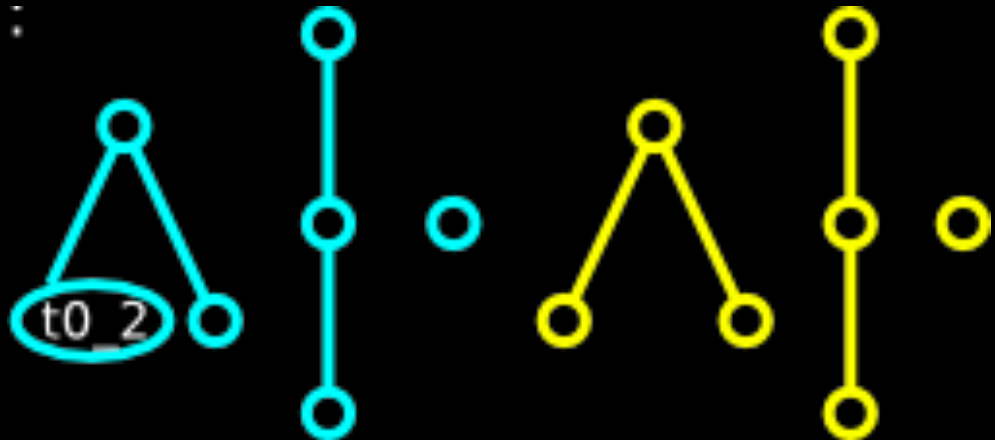
Transformations



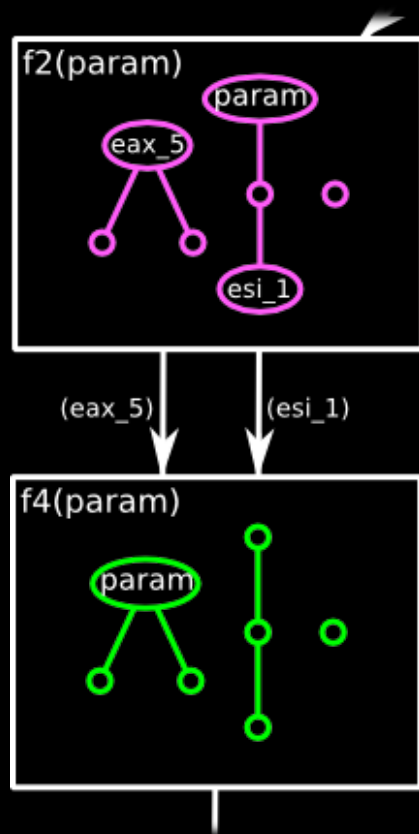
Transformations



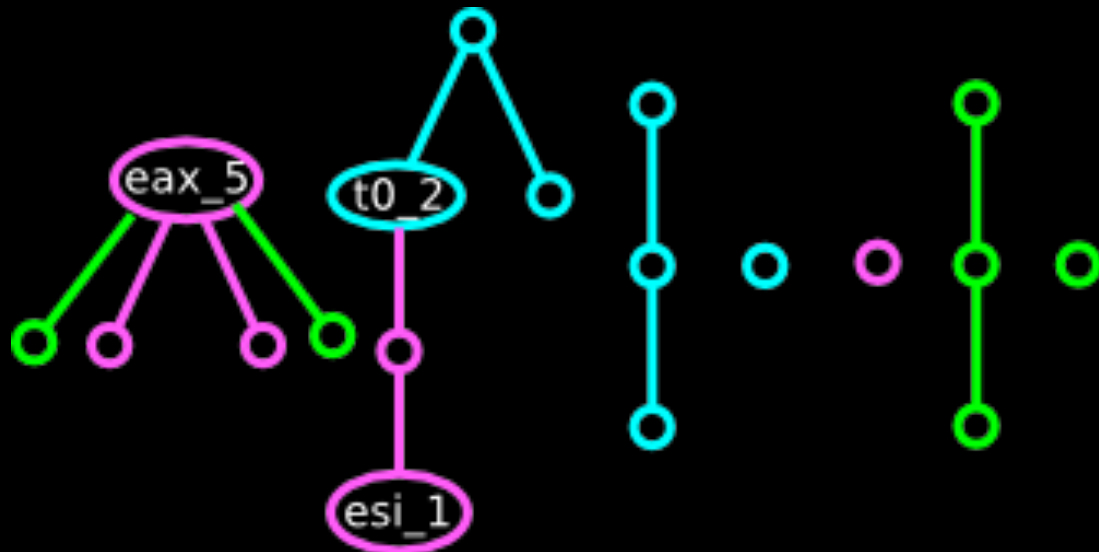
f1() \longrightarrow f3()



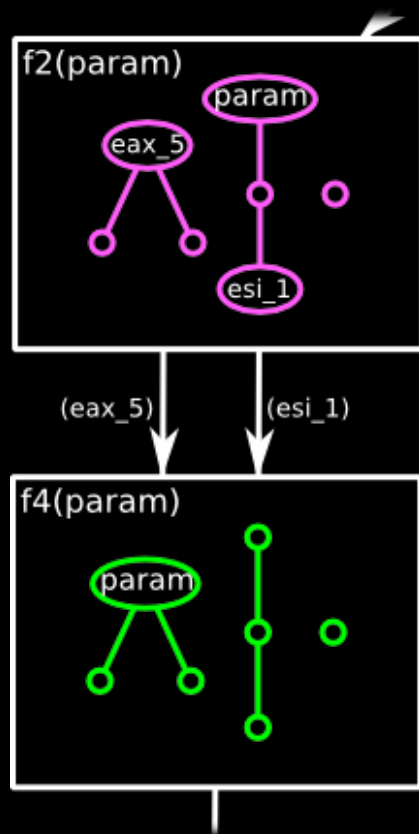
Transformations



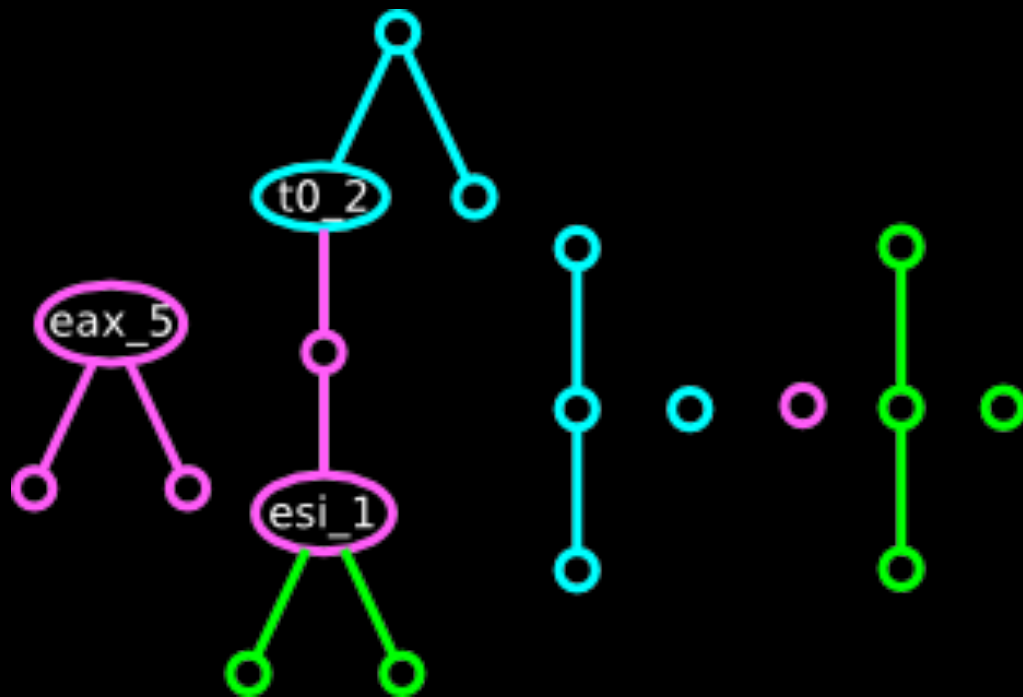
`f1()` $\xrightarrow{(t0_2)}$ `f2(param)` $\xrightarrow{(eax_5)}$ `f4(param)`



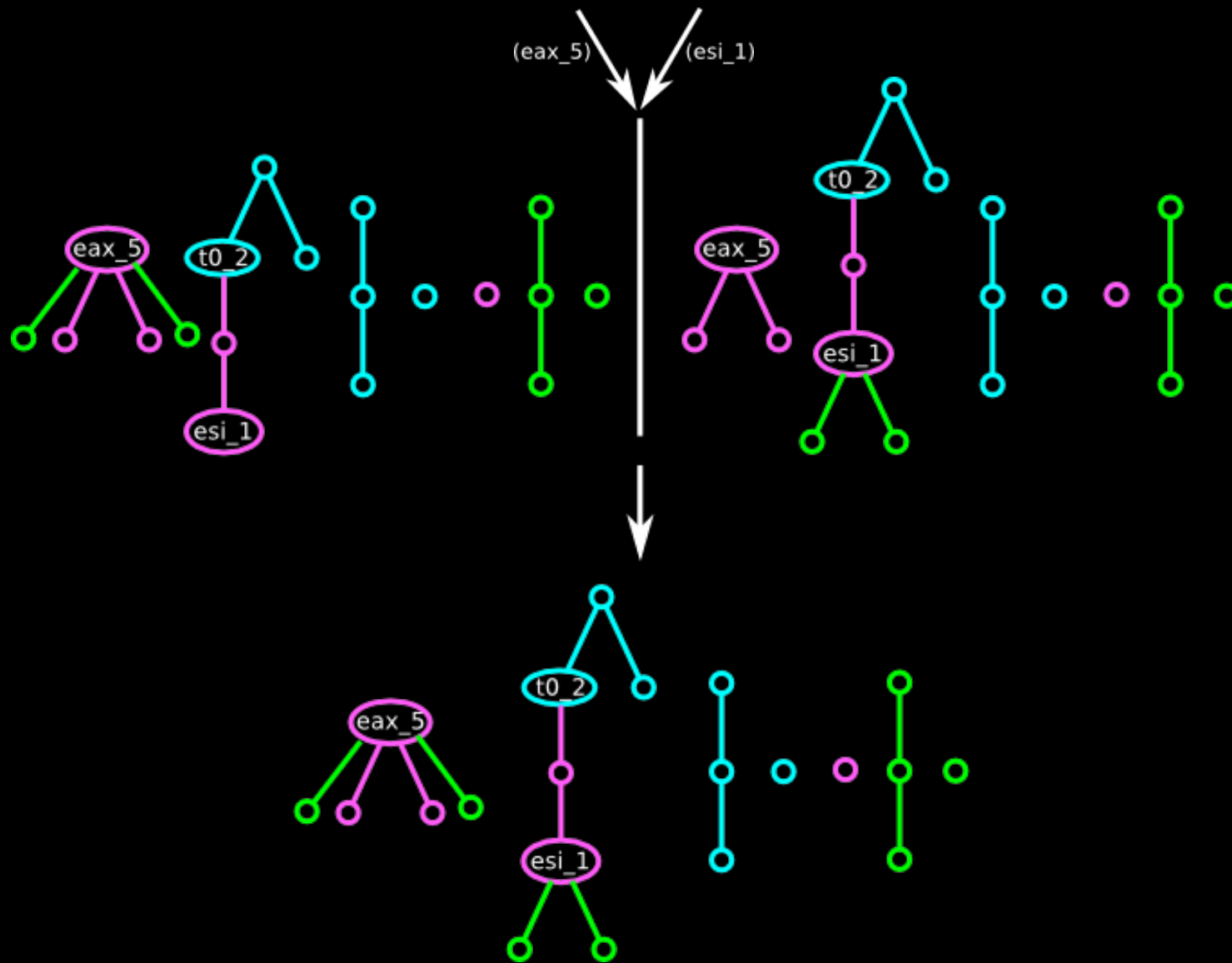
Transformations



`f1()` $\xrightarrow{(t0_2)}$ `f2(param)` $\xrightarrow{(esi_1)}$ `f4(param)`



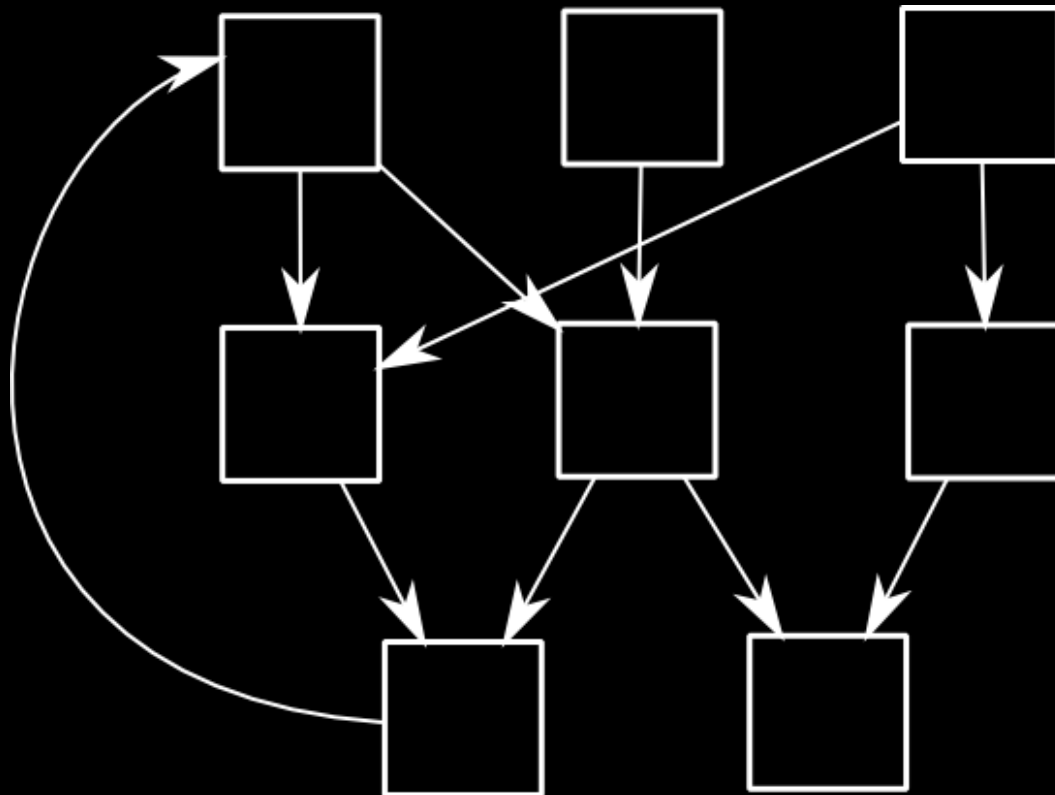
combine()



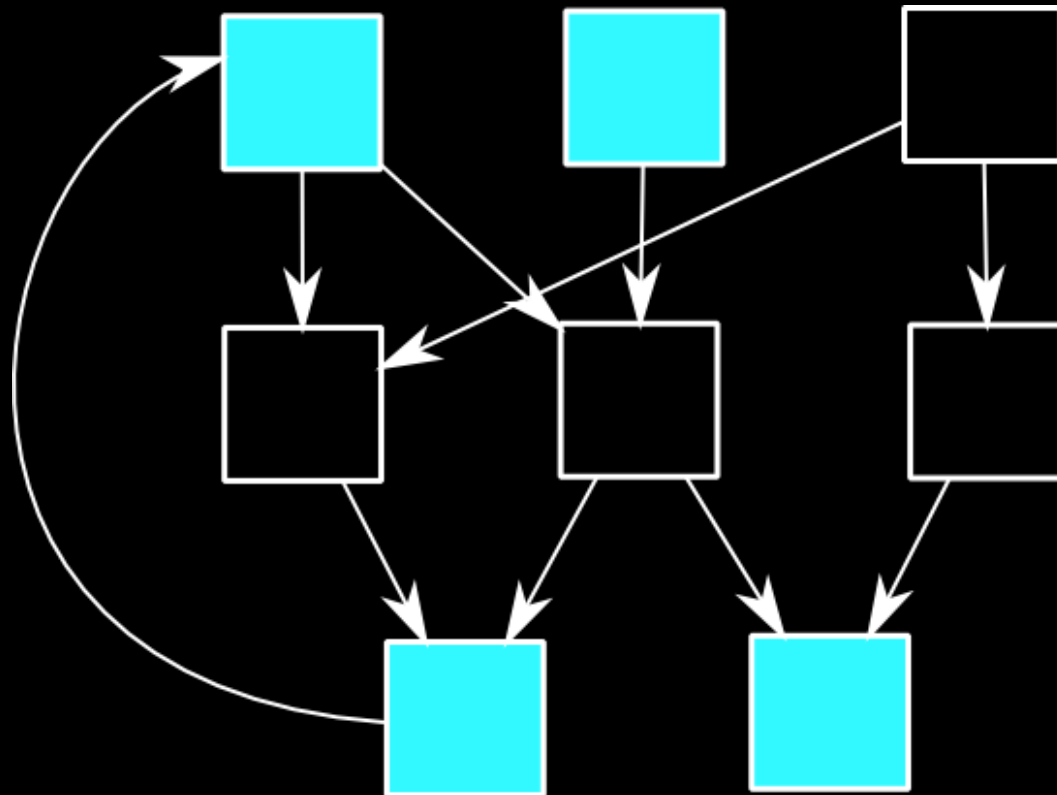
Bug Detection



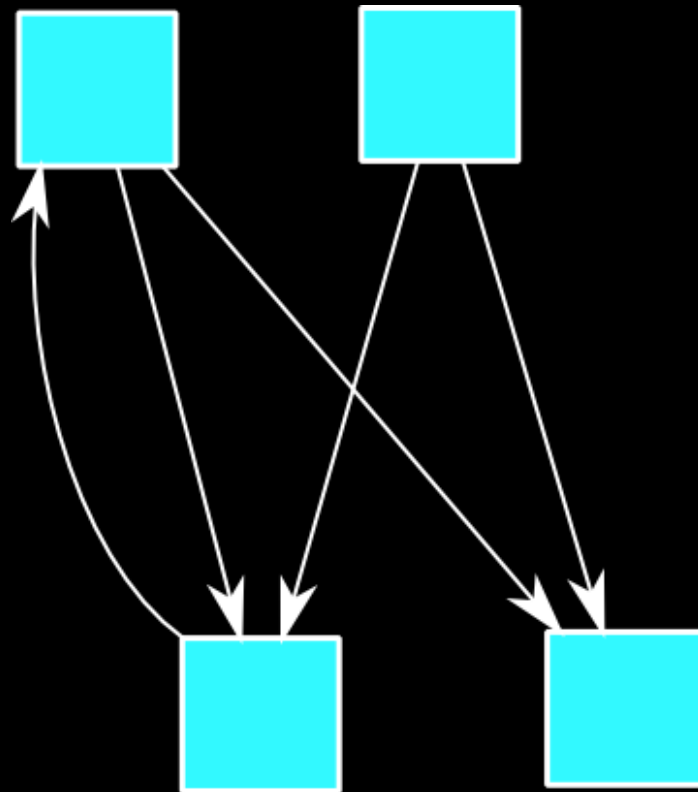
Callgraph pruning



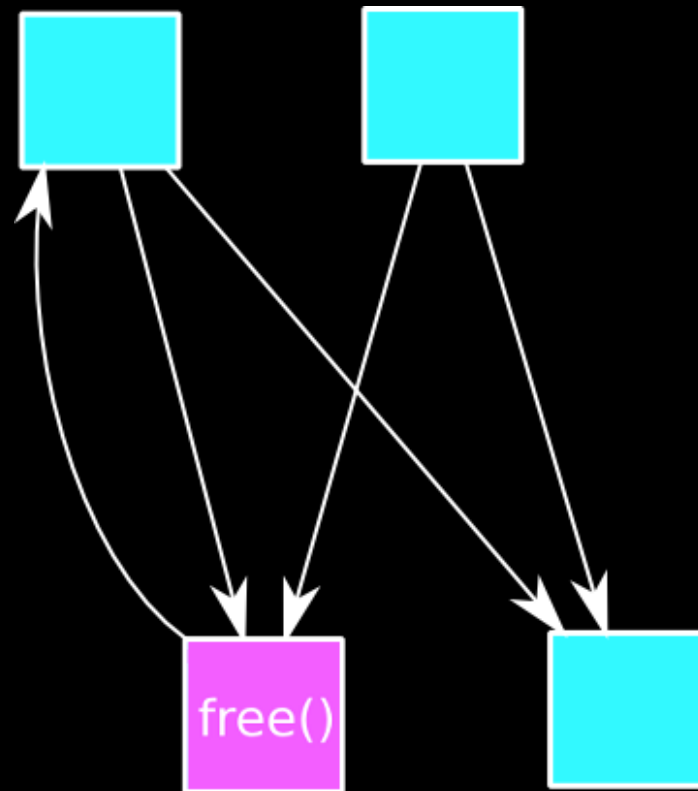
Callgraph pruning



Callgraph pruning



Marking destructor() calls

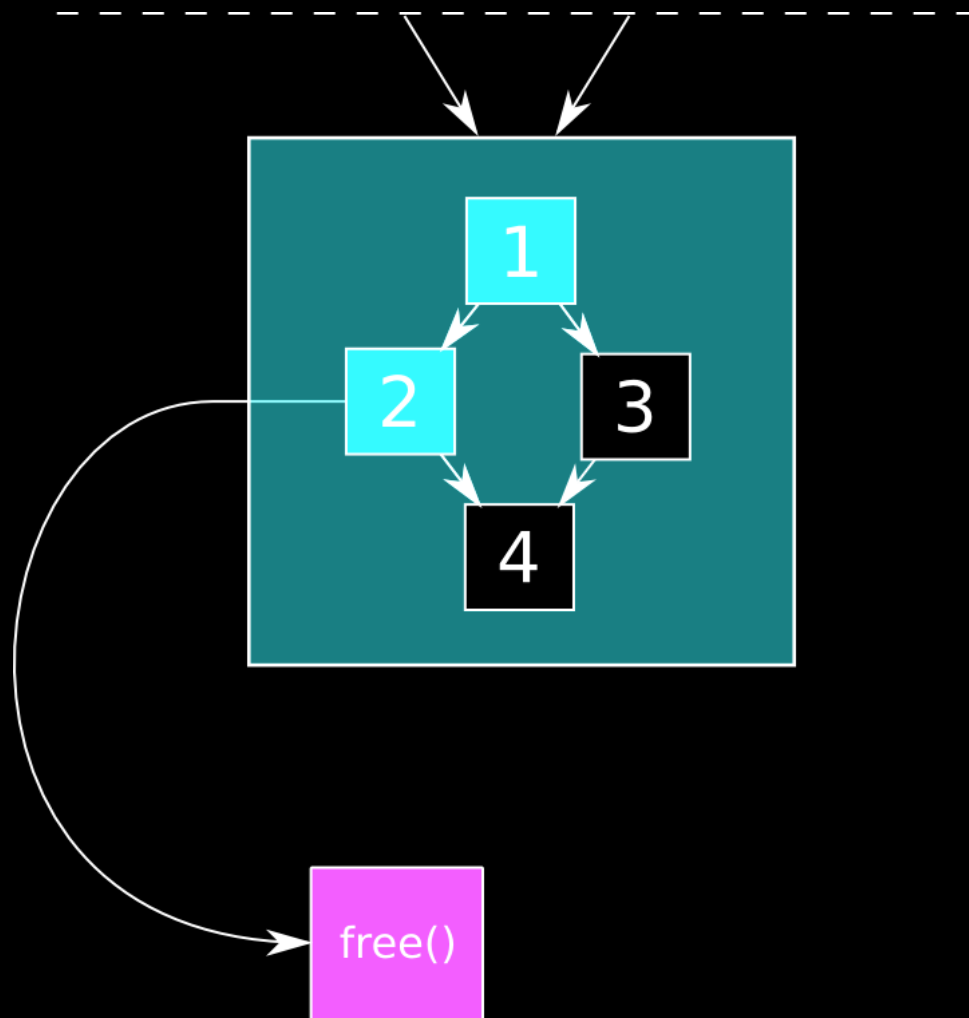


Algorithm

- v is a tracked alias
- X is a basic block of F that calls the destructor
- B is a basic block of F that accesses v or calls a function that accesses v
- Verify the following:
 - if $B \in \text{dom}(X) \Rightarrow v$ is a stale pointer
 - if $B \notin \text{dom}(X) \wedge B \in \text{succ}(X) \Rightarrow v$ may be a stale pointer
 - if $X \notin \text{dom}(B) \wedge X \in \text{succ}(B) \Rightarrow v$ may cause memory leak
 - if $X \notin \text{dom}(B) \wedge X \notin \text{succ}(B) \Rightarrow v$ causes memory leak
- Iterate substituting:
 - F with each of its callers
 - X with a basic block that calls F

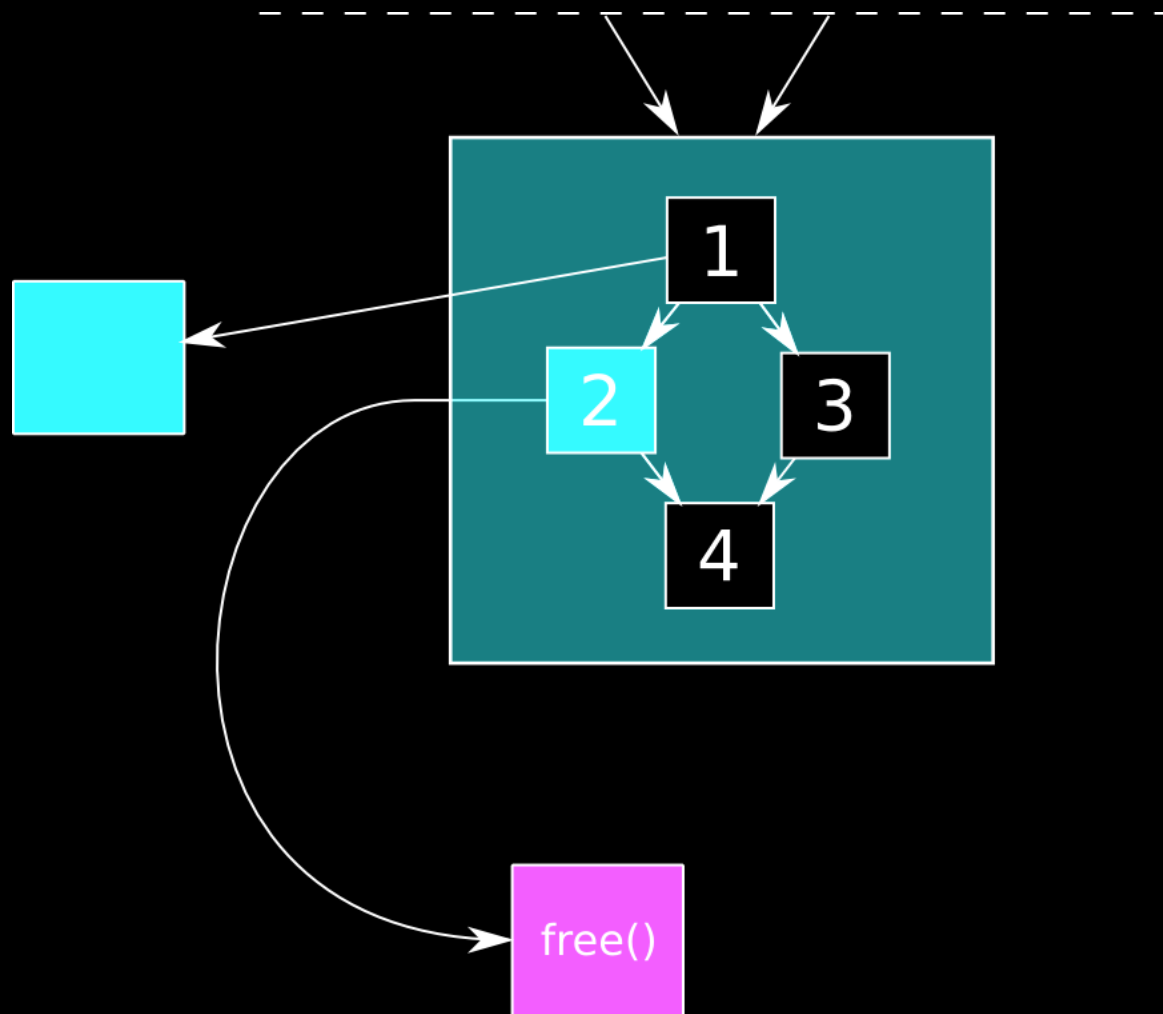
Example

No bugs



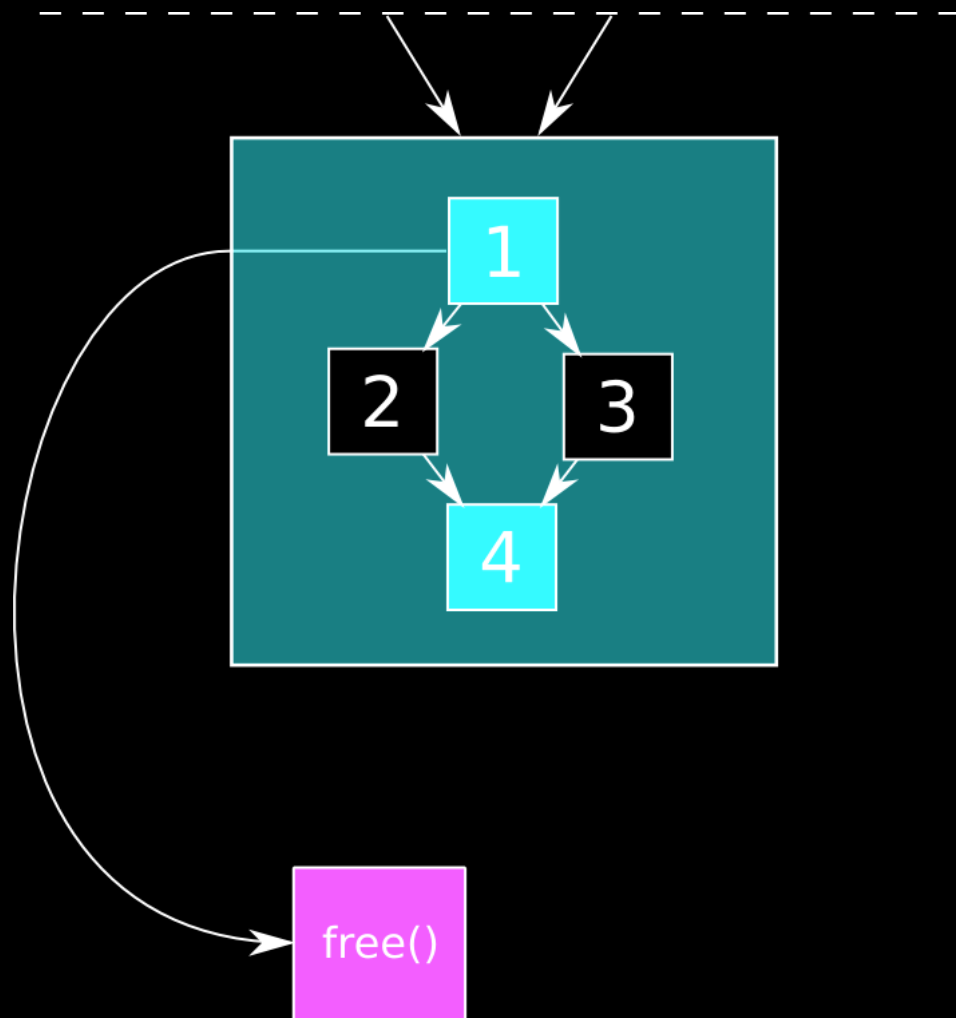
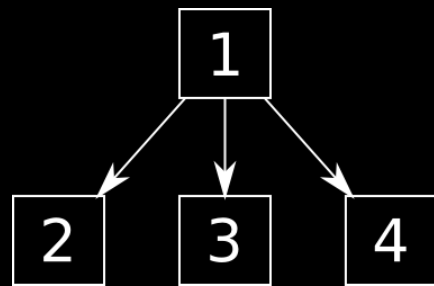
Example

No bugs



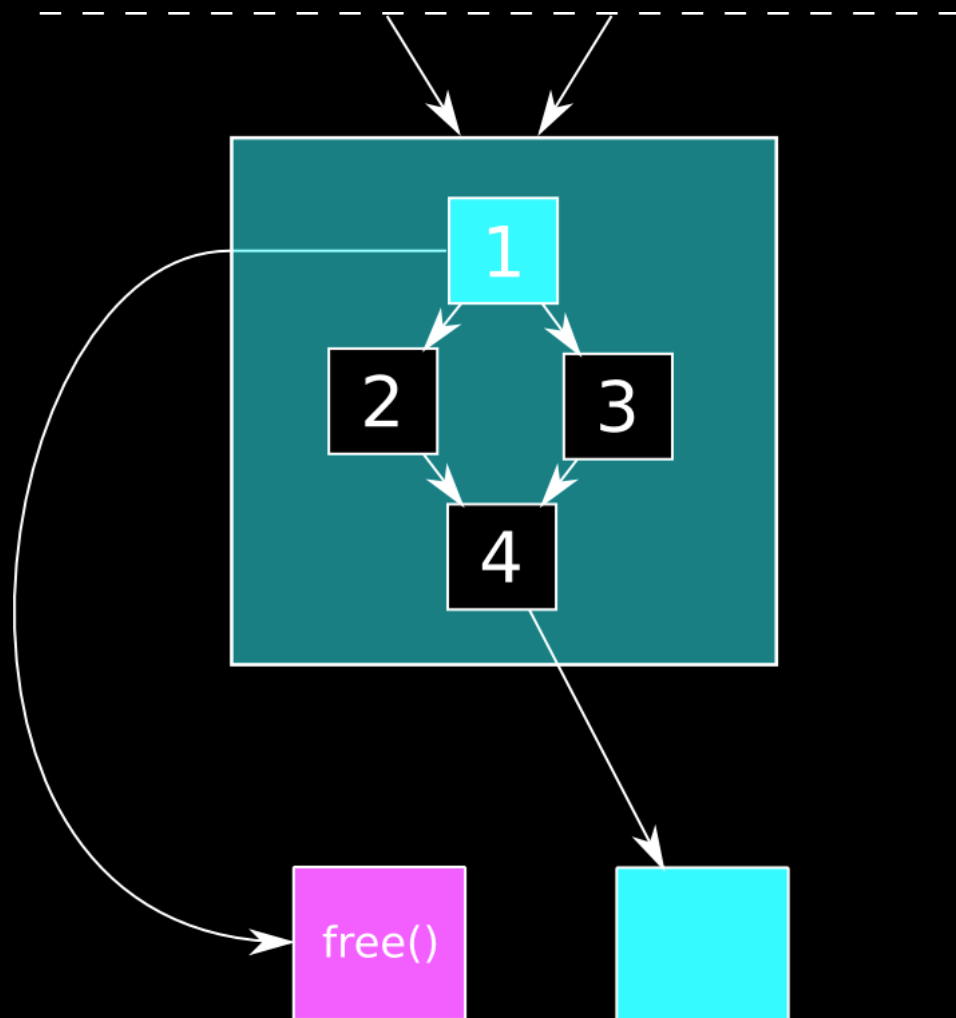
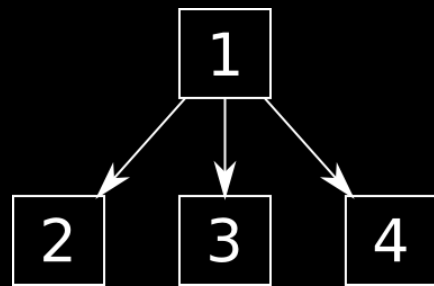
Example

Use after free bug



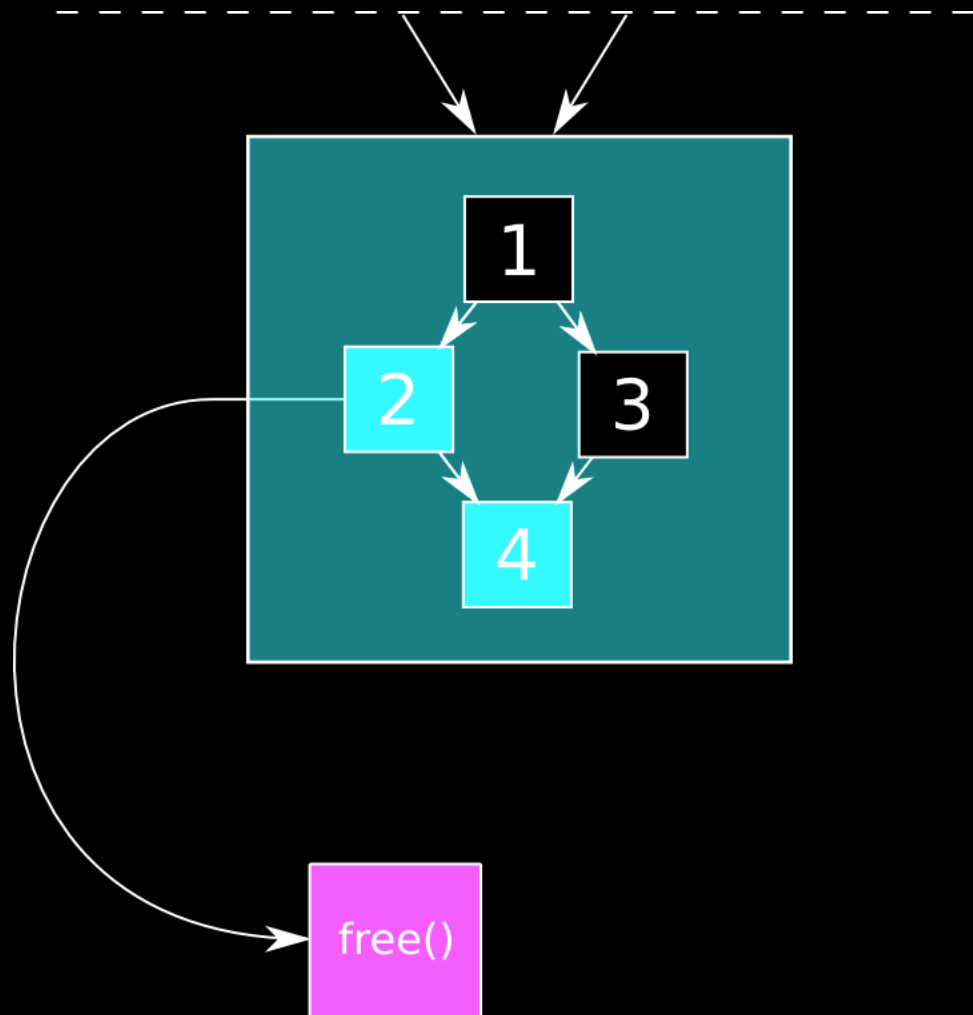
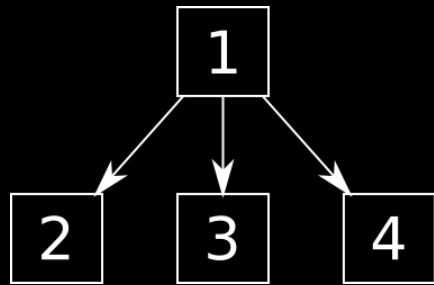
Example

Use after free bug



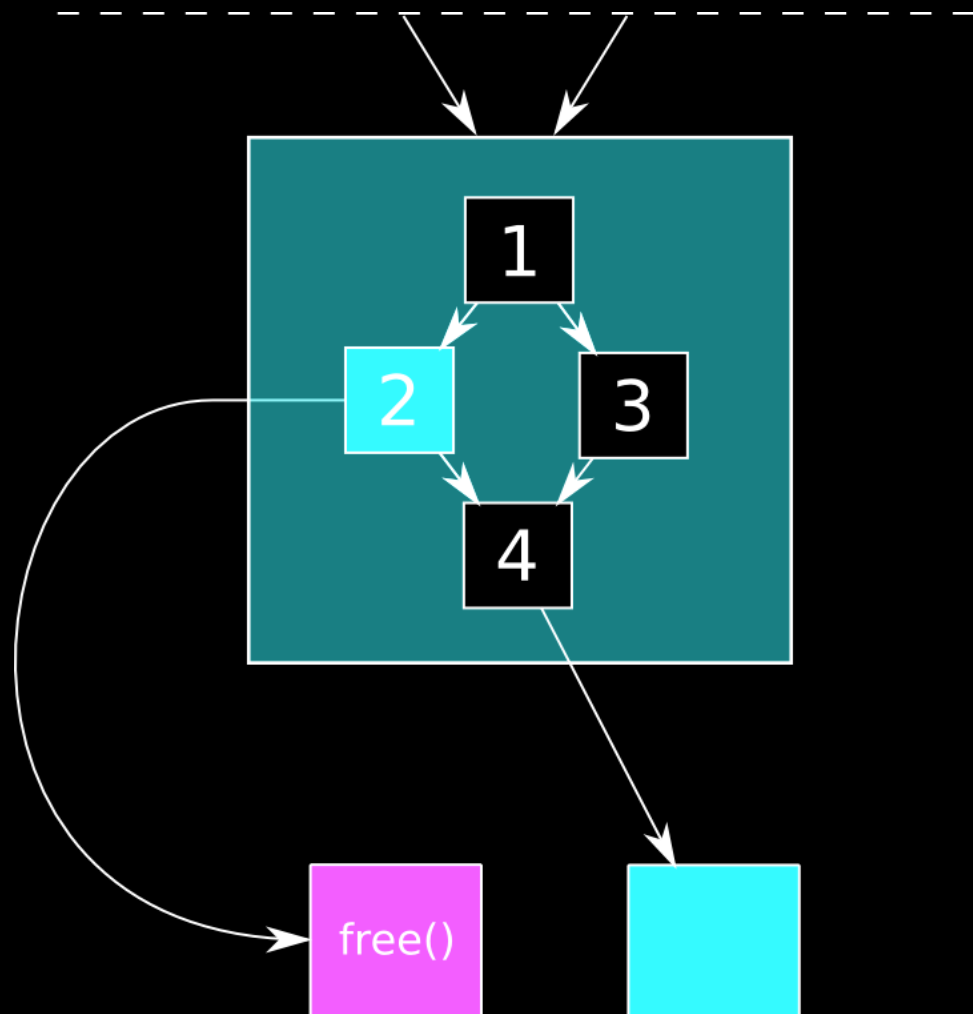
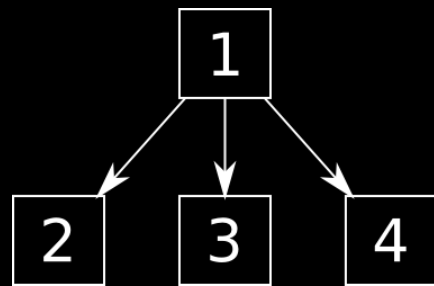
Example

Use after free bug (maybe)



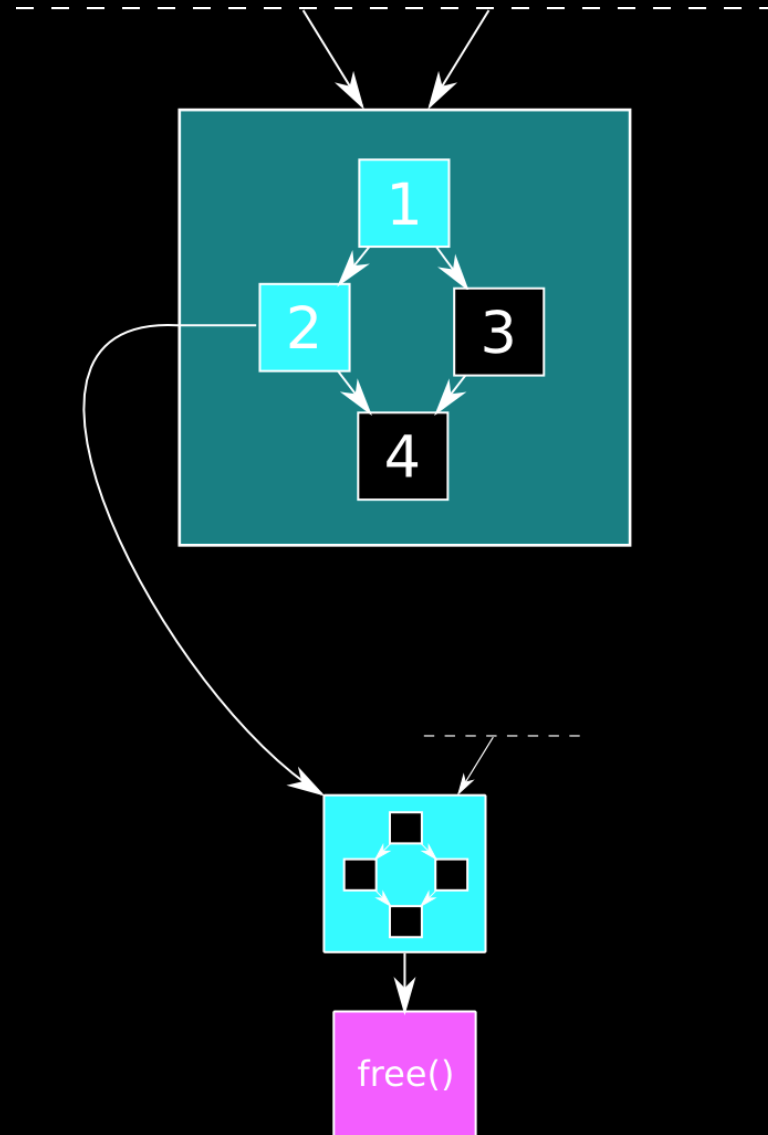
Example

Use after free bug (maybe)



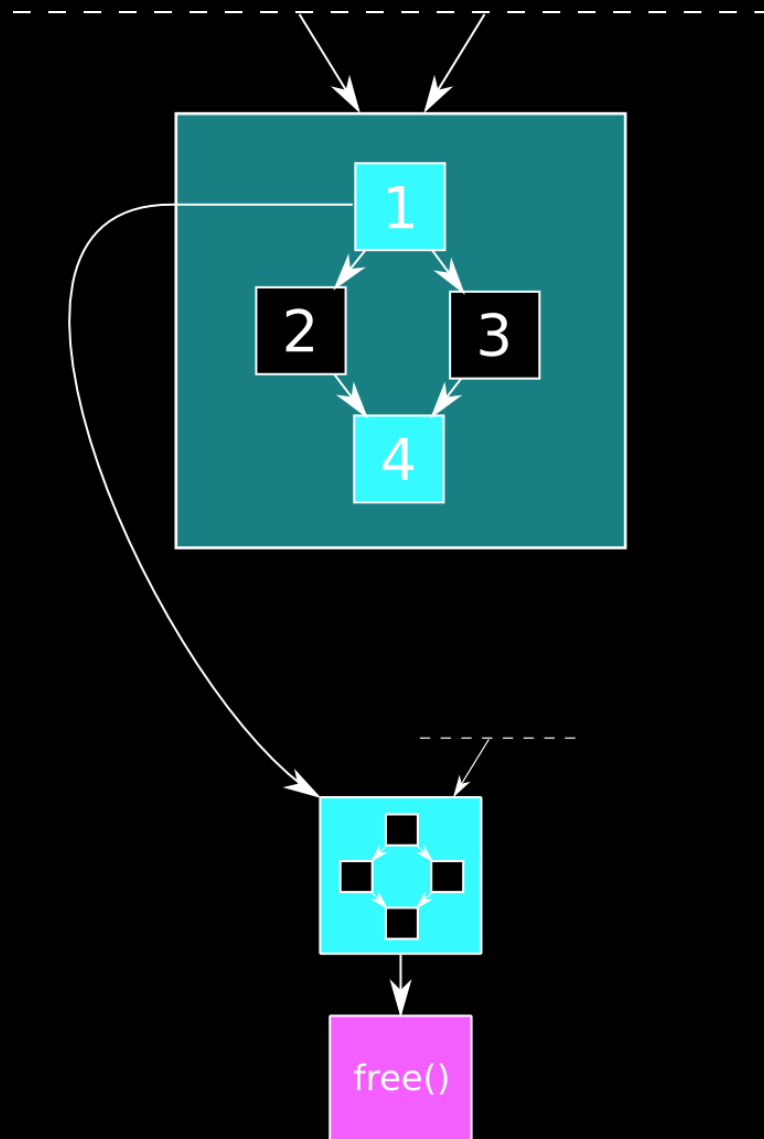
Example

No bugs



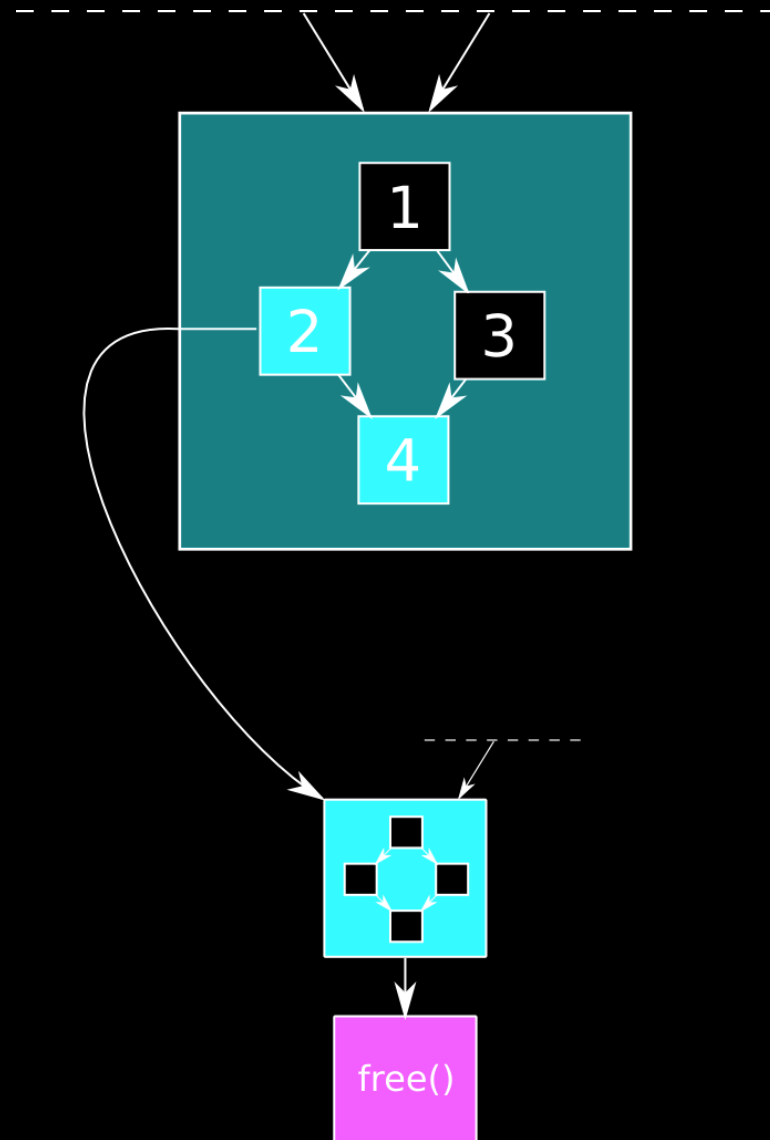
Example

Use after free bug



Example

Use after free bug (maybe)



What's the catch

- We cannot handle all data structures
- We cannot handle function pointers
- We have false positives
- We have false negatives
- Some “smart pointers”-like interfaces might not be covered
- The best use is for C++ life-span issues

Future

- Increase the number of covered data structure
- Use a solver to reduce false positives
- Import dynamic analysis data to mitigate the function pointers problem

That's all folks

