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 [illegible]tion, code snippets and bullets.

From past experiences the speak[illegible]t all the aforementioned elements are no useful i[illegible]ple understand your idea.

You instead will see a[illegible]pictures which the speaker hopes will convey bett[illegible]anding of the ideas explained in the talk

$$(S_N f)(x) = \frac{a_0}{2} + \sum_{n=1}^{N} [a_n \cos(nx) + b_n \sin(nx)], \quad N \geq 0$$

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

Stale pointers are the new black

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.

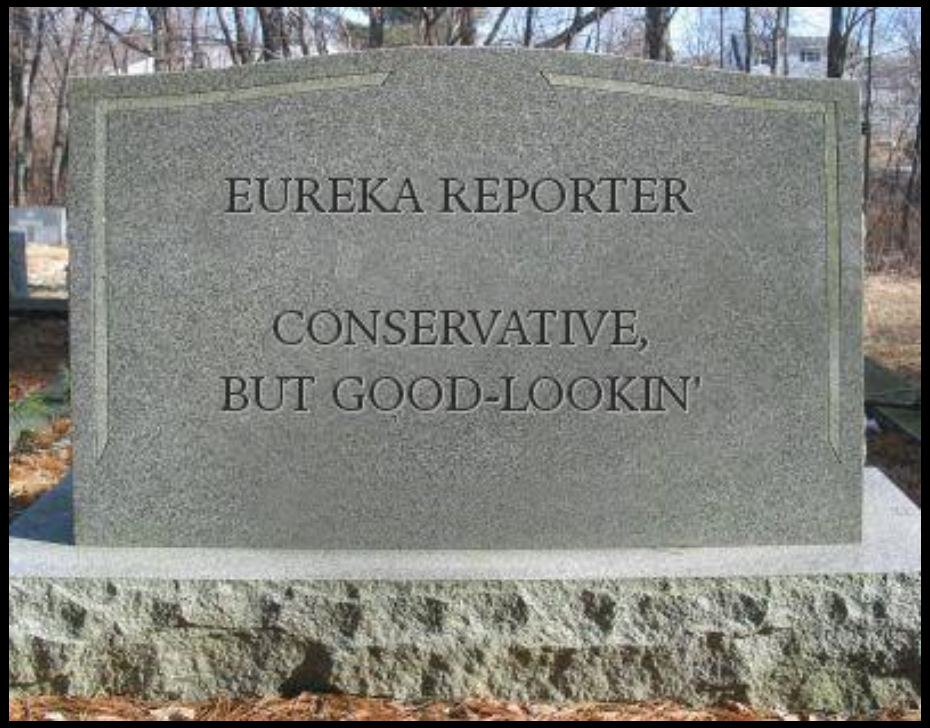
17 Aug via web ☆ Favorite ↻ Retweet ↩ Reply

Different approaches then..

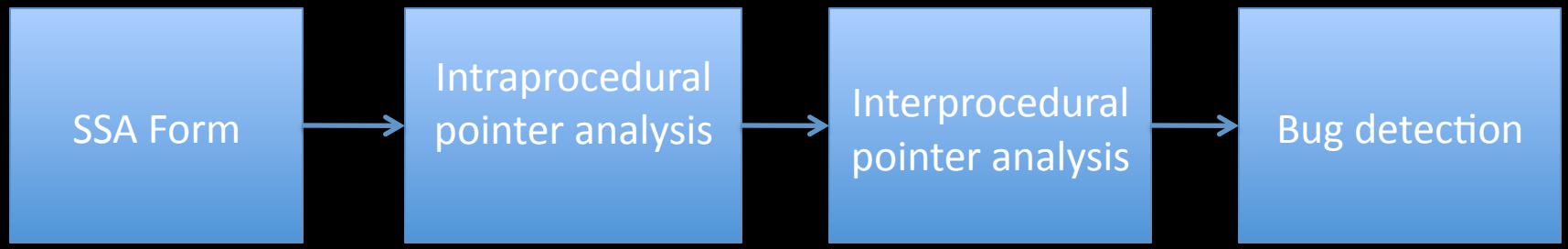
.. Or Static Analysis!

- Dataflow analysis
- Model Checking
- Theorem Proving

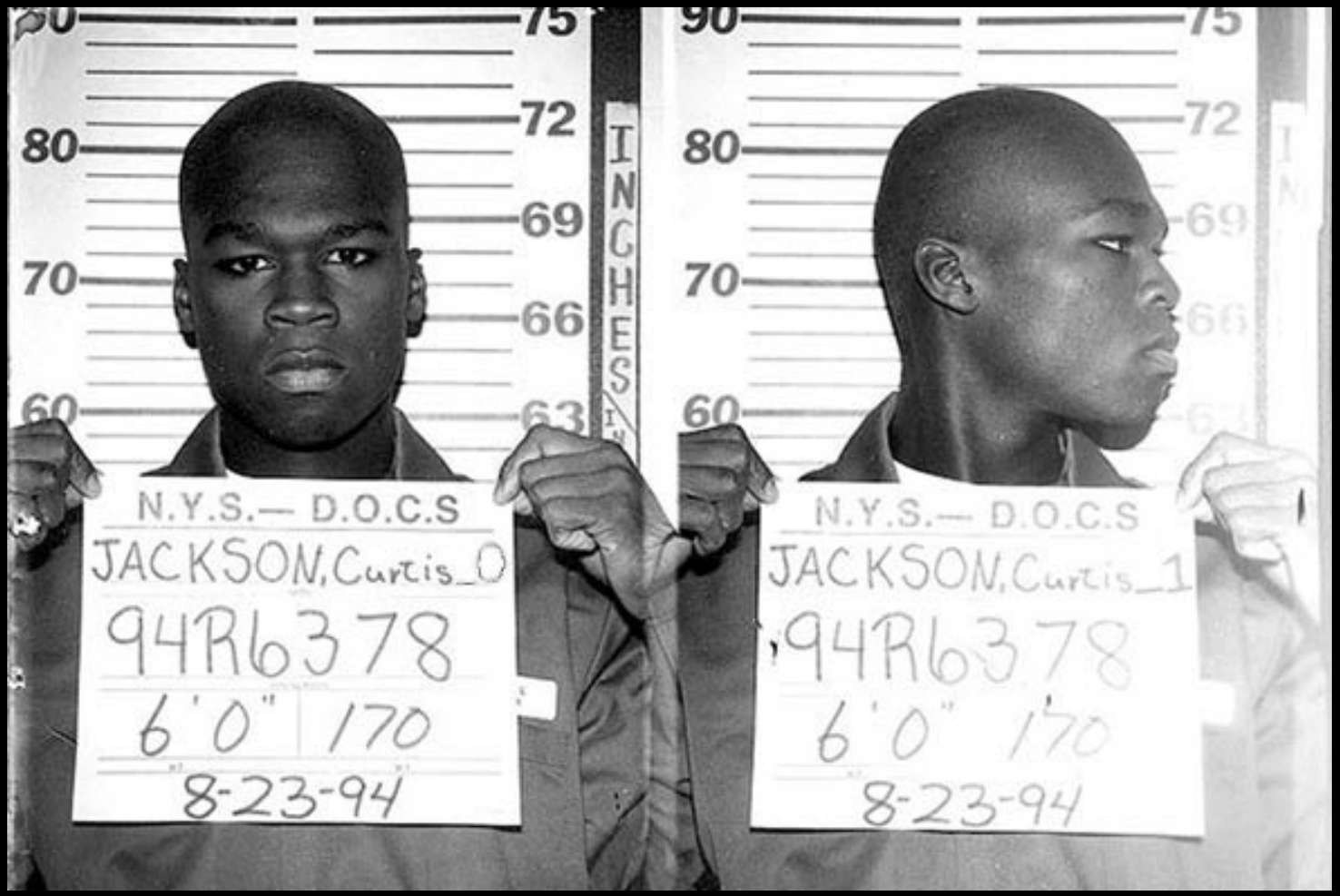
Our Idea



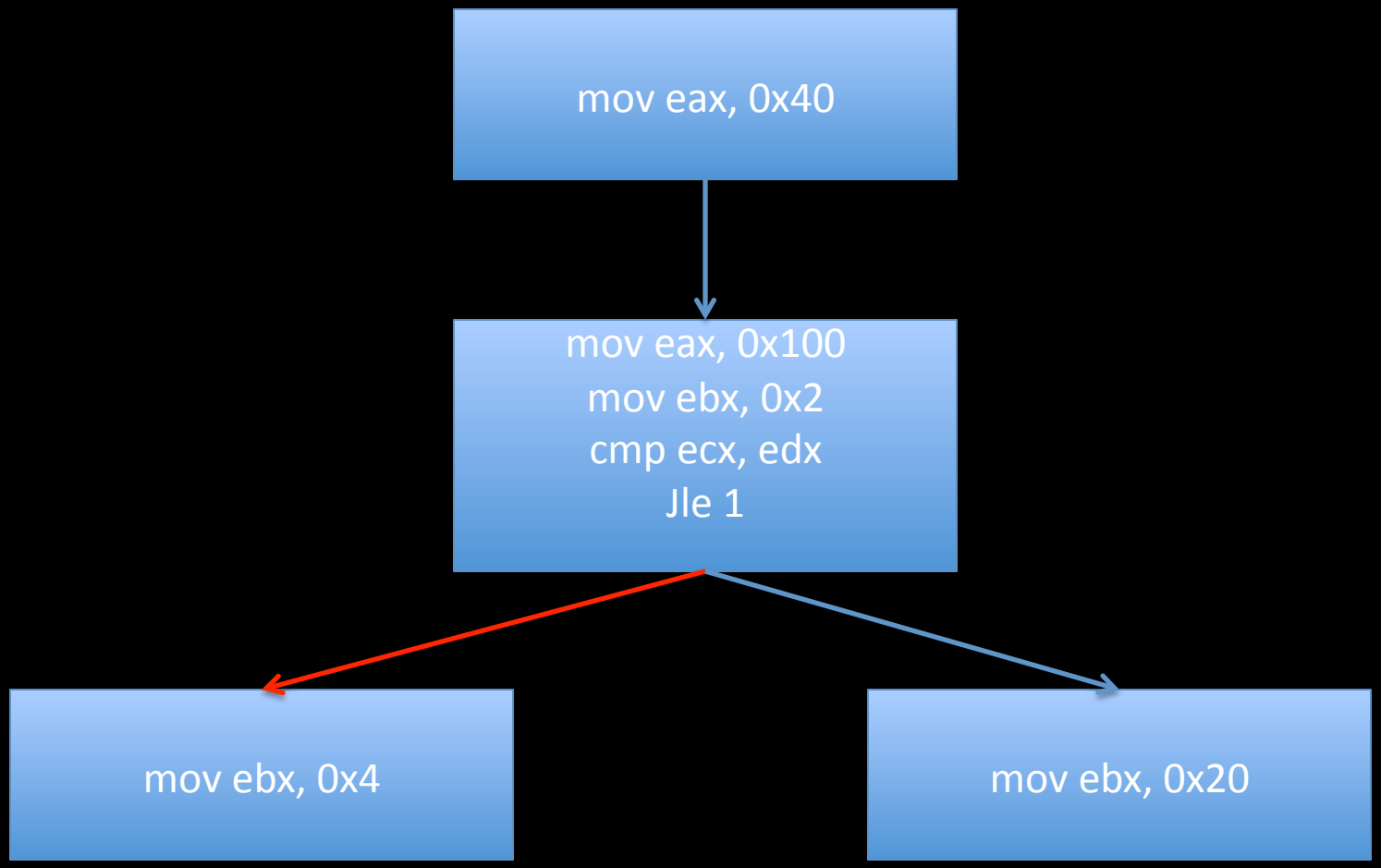
How it works


SSA Form
Intraprocedural pointer analysis
Interprocedural pointer analysis
Bug detection

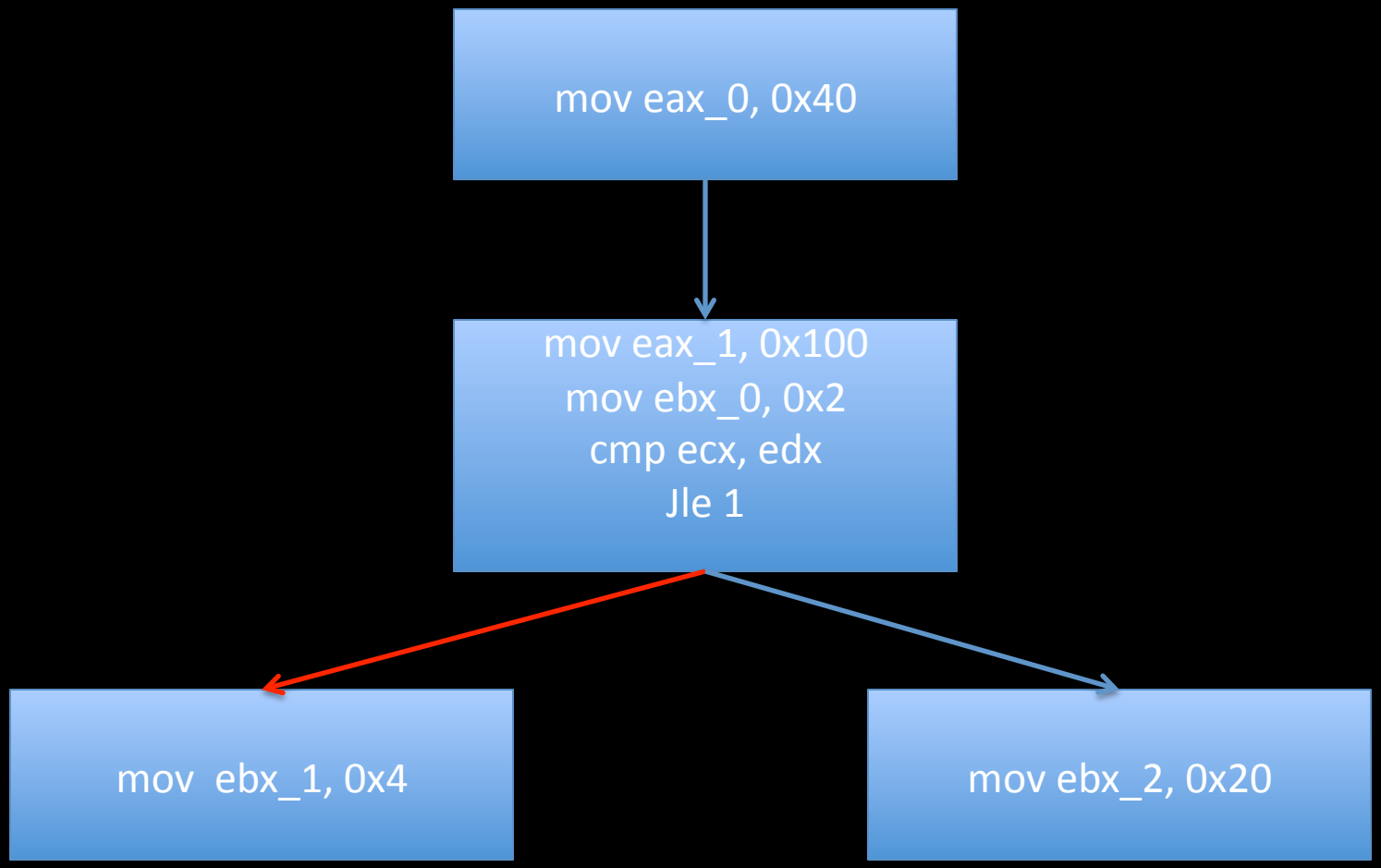
Single Static Assignment Form



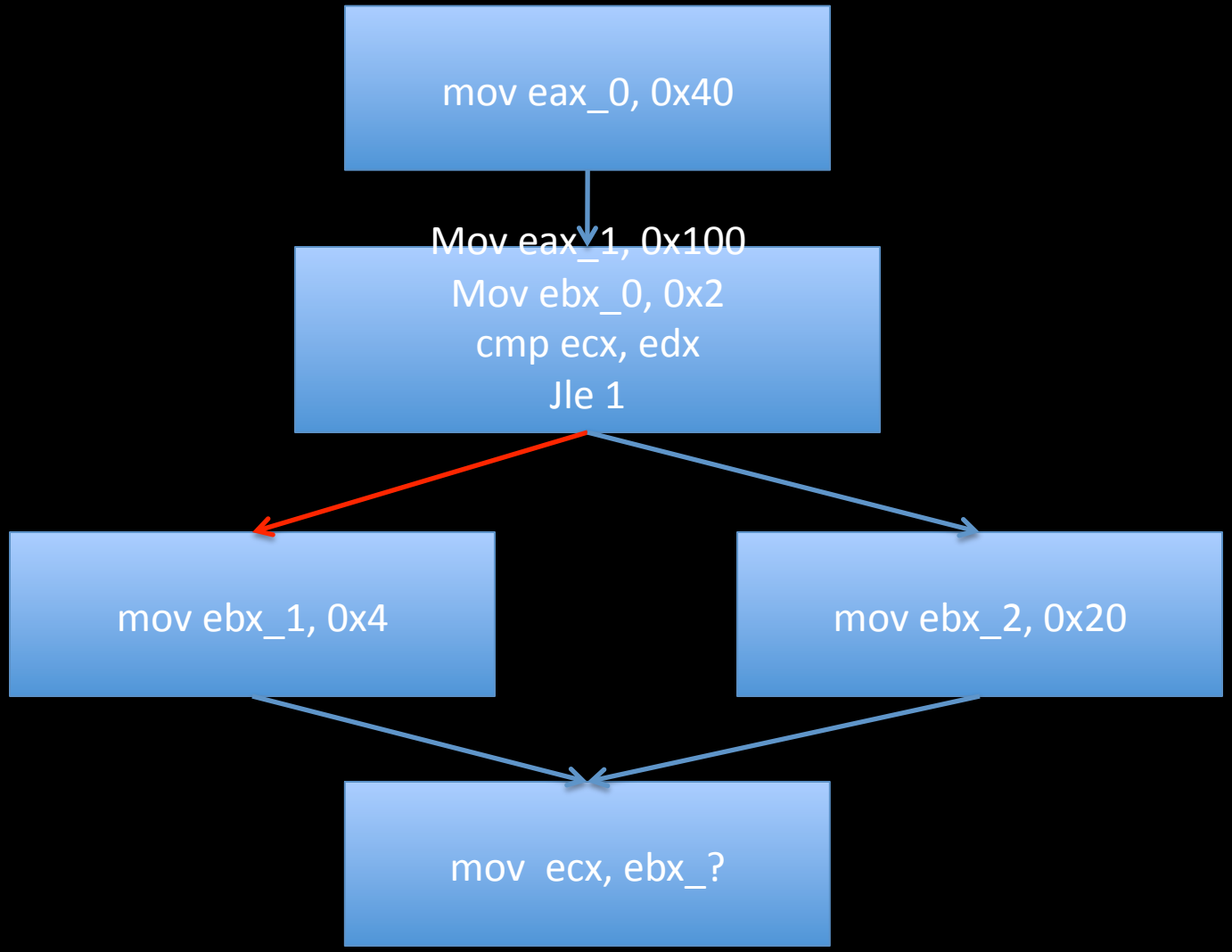
Example


mov eax, 0x40
mov eax, 0x100
mov ebx, 0x2
cmp ecx, edx
Jle 1
mov ebx, 0x4
mov ebx, 0x20

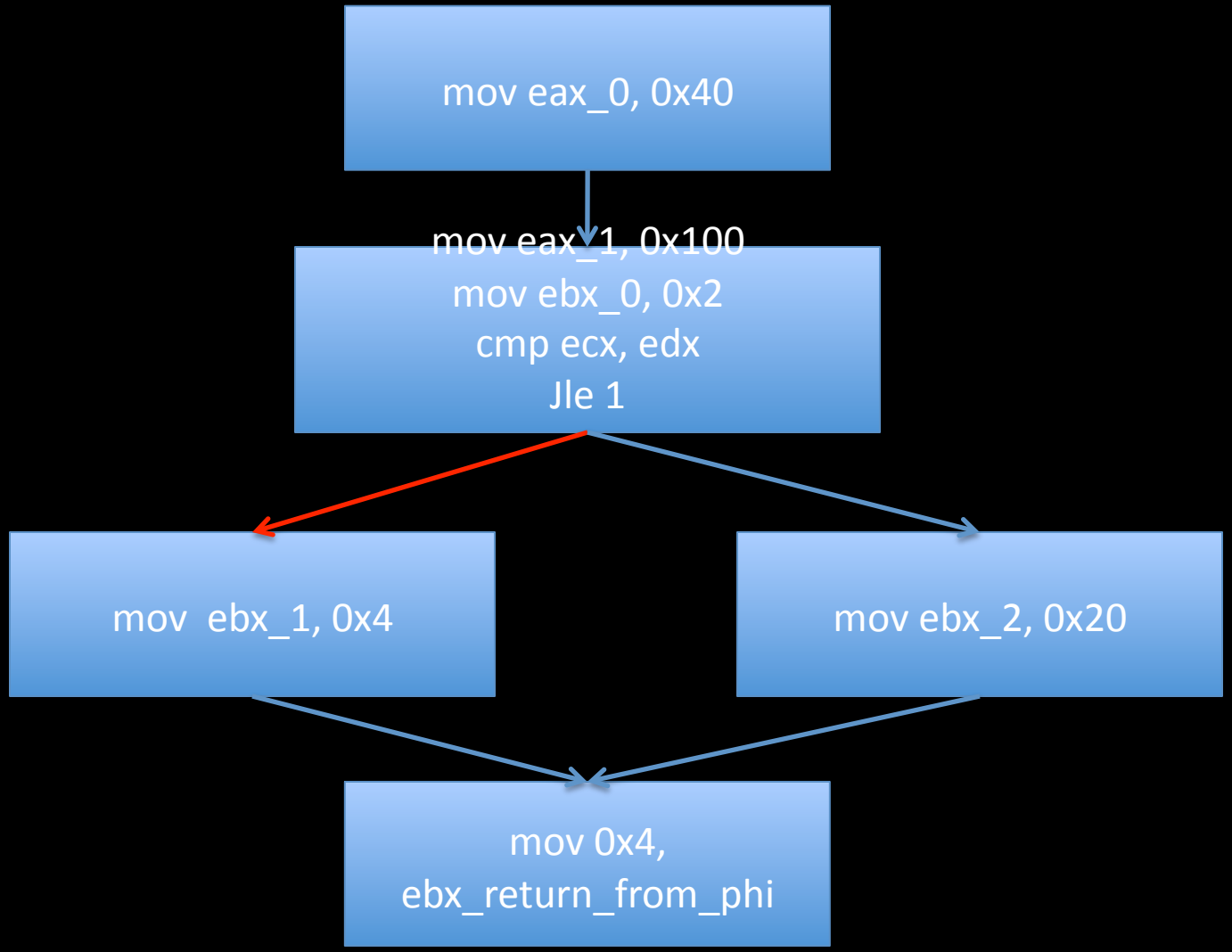
Looks better, right?



How about now?



Tah-dah!


mov eax_0, 0x40
mov eax_1, 0x100
mov ebx_0, 0x2
cmp ecx, edx
Jle 1
mov ebx_1, 0x4
mov ebx_2, 0x20
mov 0x4,
ebx_return_from_phi

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

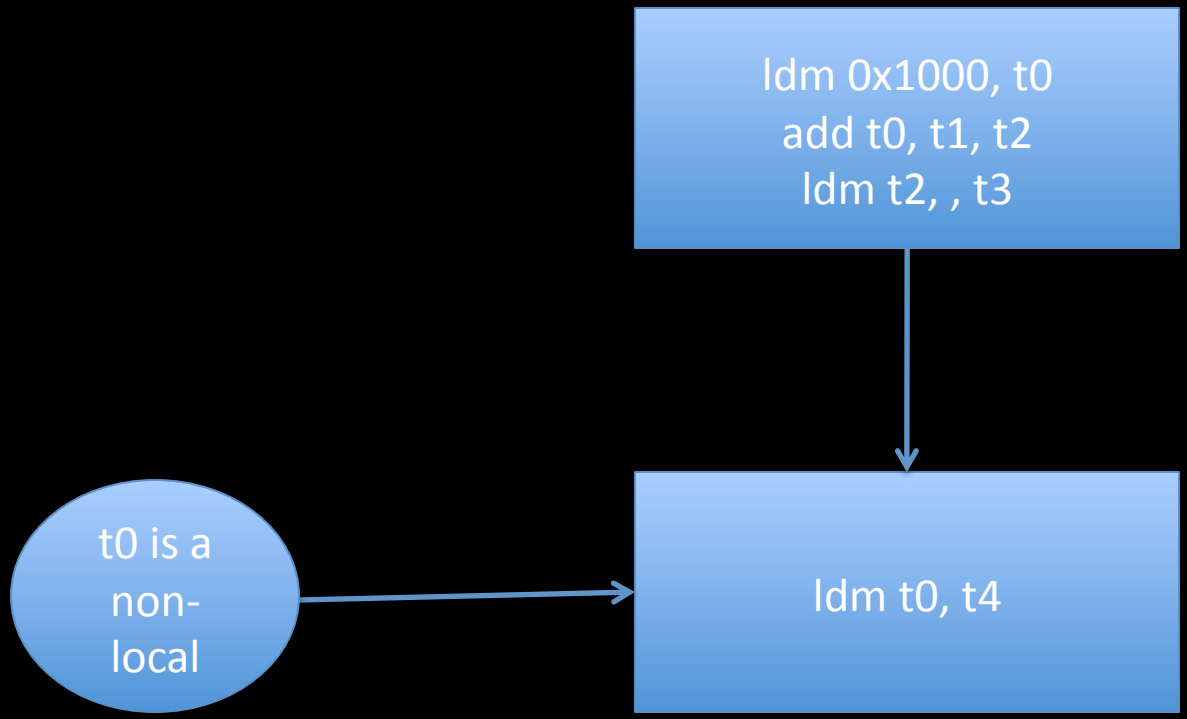
```
001EB000    sub         esp, 0x4, qword t0              // 00001EB0 push ebp
001EB001    and         qword t0, 0xFFFFFFFF, esp
001EB002    stm         ebp,  , esp
001EB100    str         esp,  , ebp                     // 00001EB1 mov ebp, esp
001EB300    add         0x8, ebp, qword t0              // 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



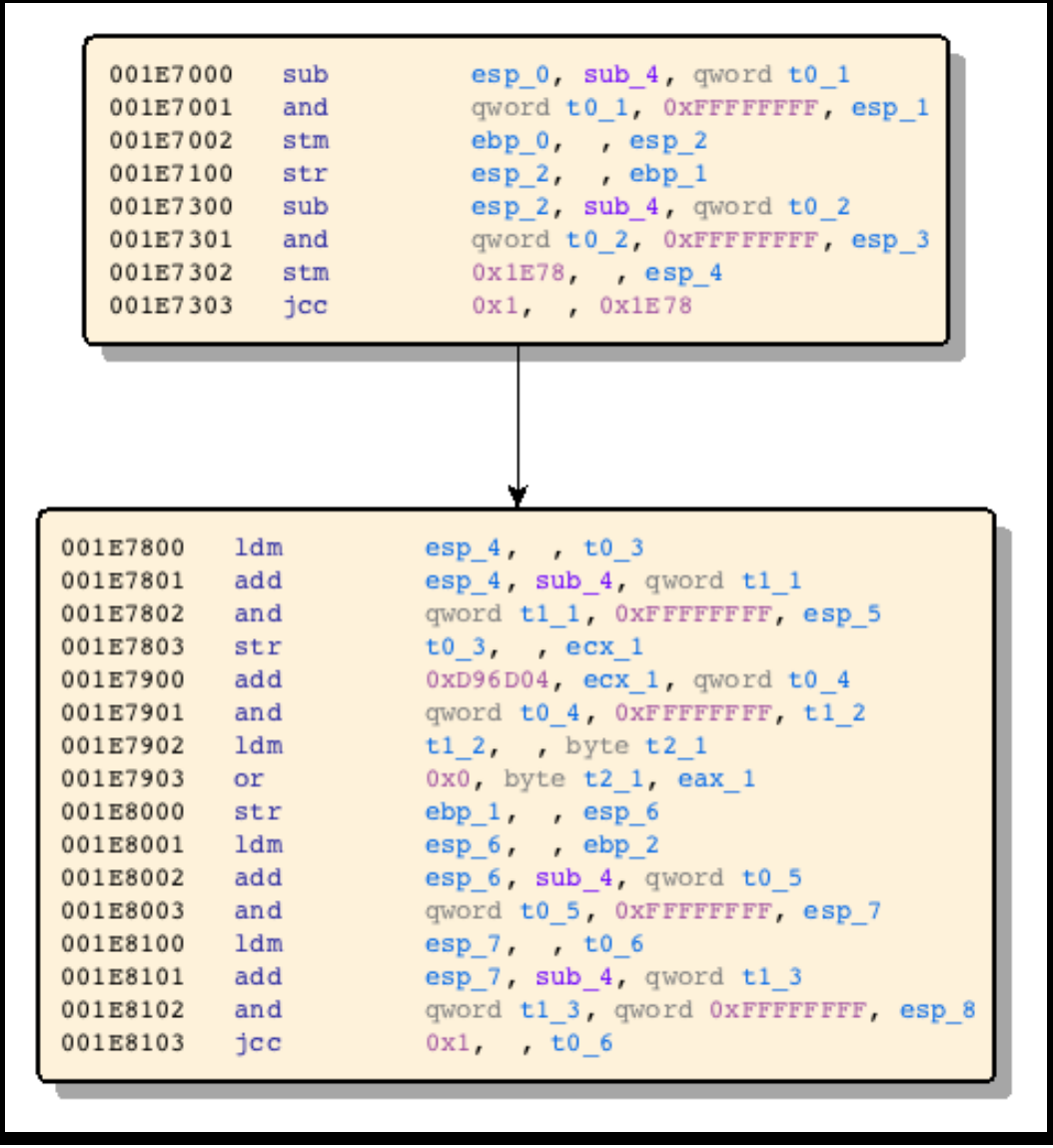
Algorithm

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

A function

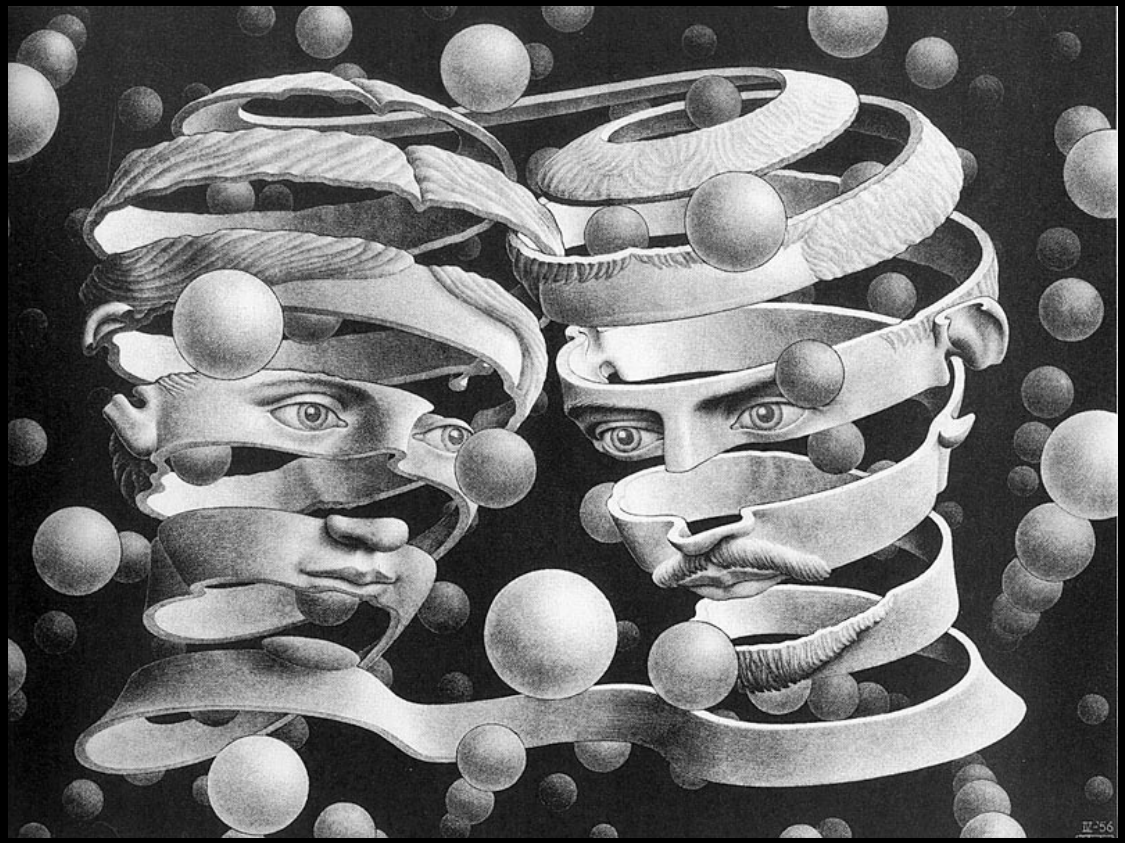

001E7000 sub esp, 0x4, qword t0 // 00001E70 push ebp
001E7001 and qword t0, 0xFFFFFFFF, esp
001E7002 stm ebp, , esp
001E7100 str esp, , ebp // 00001E71 mov ebp, esp
001E7300 sub esp, 0x4, qword t0 // 00001E73 call cs: 7800
001E7301 and qword t0, 0xFFFFFFFF, esp
001E7302 stm 0x1E78, , esp
001E7303 jcc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

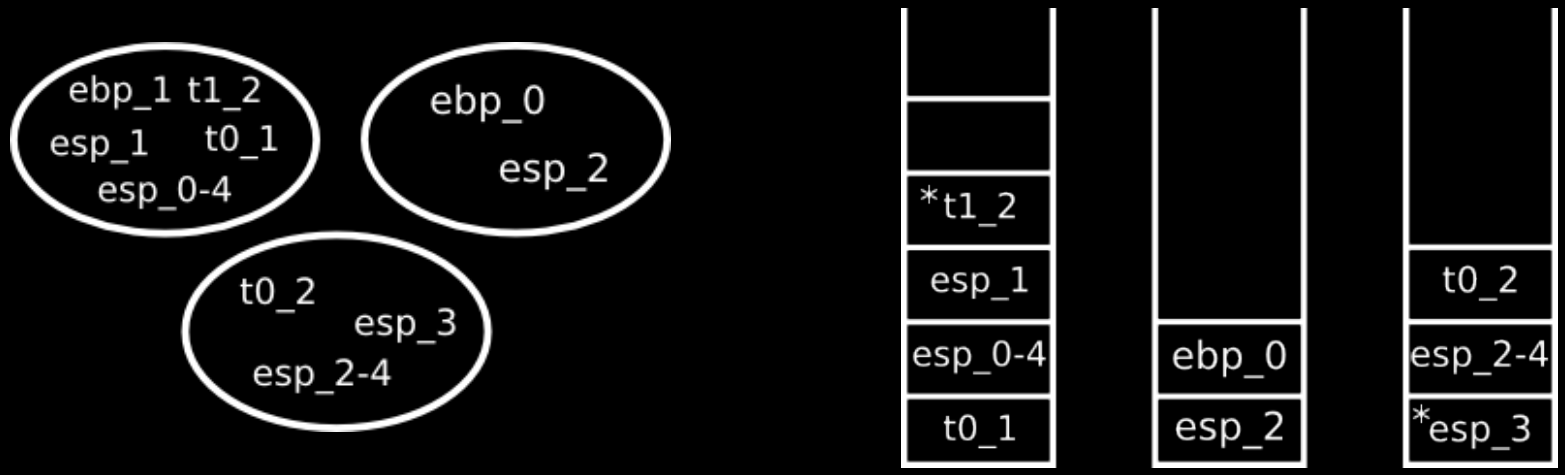
The lattice has to satisfy the ascending chain condition

Now the analysis itself

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

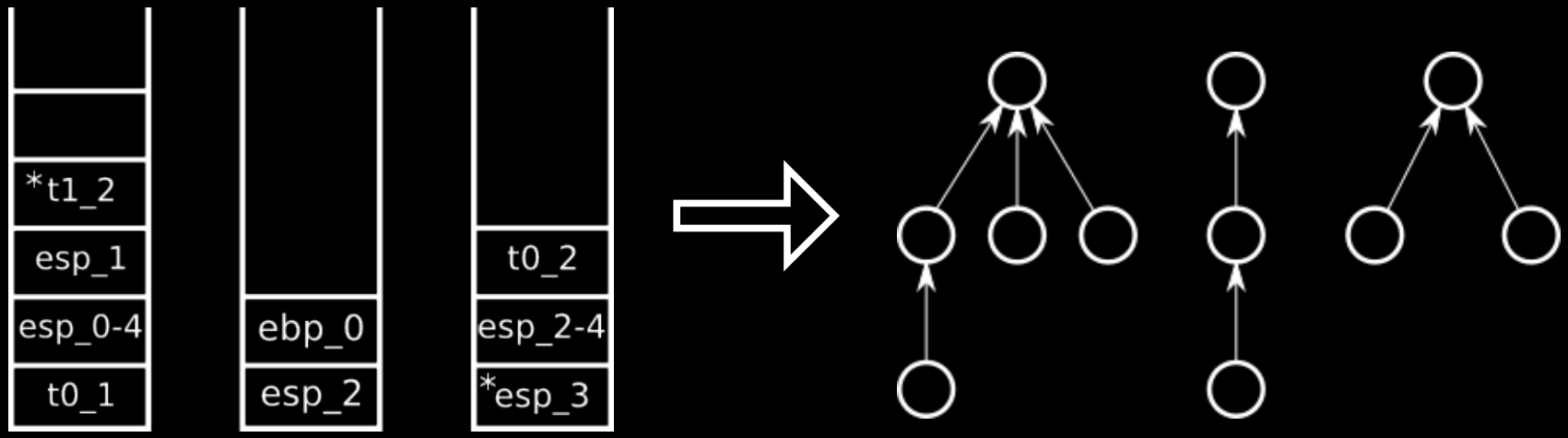
ARITHMETIC INSTRUCTIONS	OPERATION
ADD x_1, x_2, y	y is added to the alias set of $x_1 + x_2$
SUB x_1, x_2, y	y is added to the alias set of $x_1 - x_2$
MUL x_1, x_2, y	y is added to the alias set of $x_1 \cdot x_2$
DIV x_1, x_2, y	y is added to the alias set of $\left\lfloor \frac{x_1}{x_2} \right\rfloor$
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 $x_1 \mid x_2$
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 mem$[x_1]$
STM x_1, ε, y	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:
  - live-out(inst) $\subseteq$ vars(dom(inst))
  - Alias list $\cap$ vars(sdom(Φ)):
    - pop() from the list until

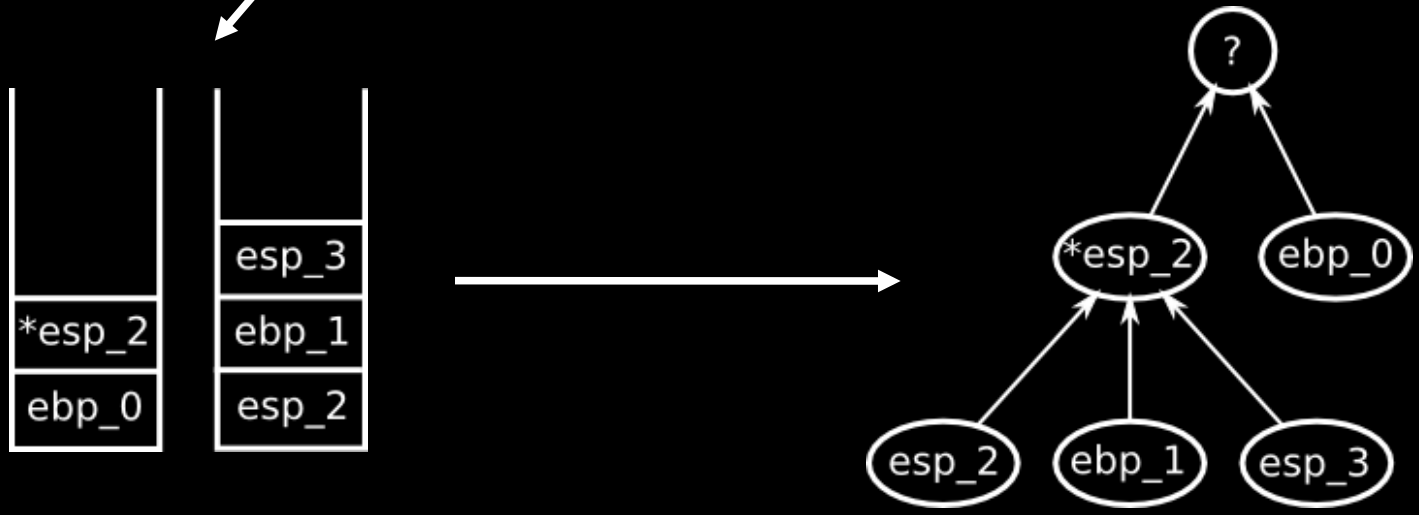
      top(alias list) $\in$ vars(sdom(Φ))
- Add aliases defined by Φ functions
- Unite the sets of lists

Data Structures Again



Example

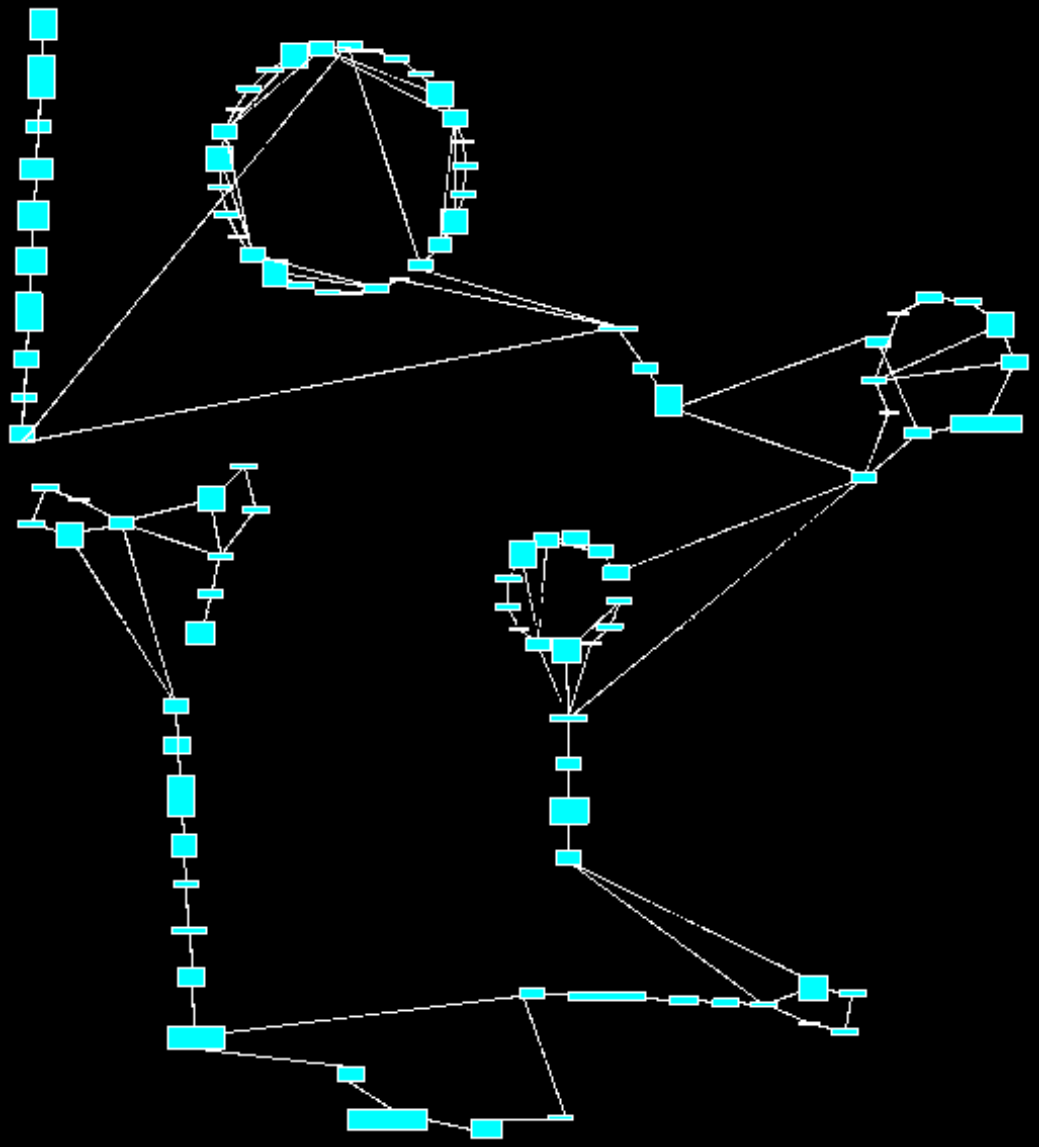
```
001ED002    stm         ebp_0,  , esp_2
001ED100    str         esp_2,  , ebp_1

001ED900    str         ebp_1,  , esp_3
```



Tracking parameters and return

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

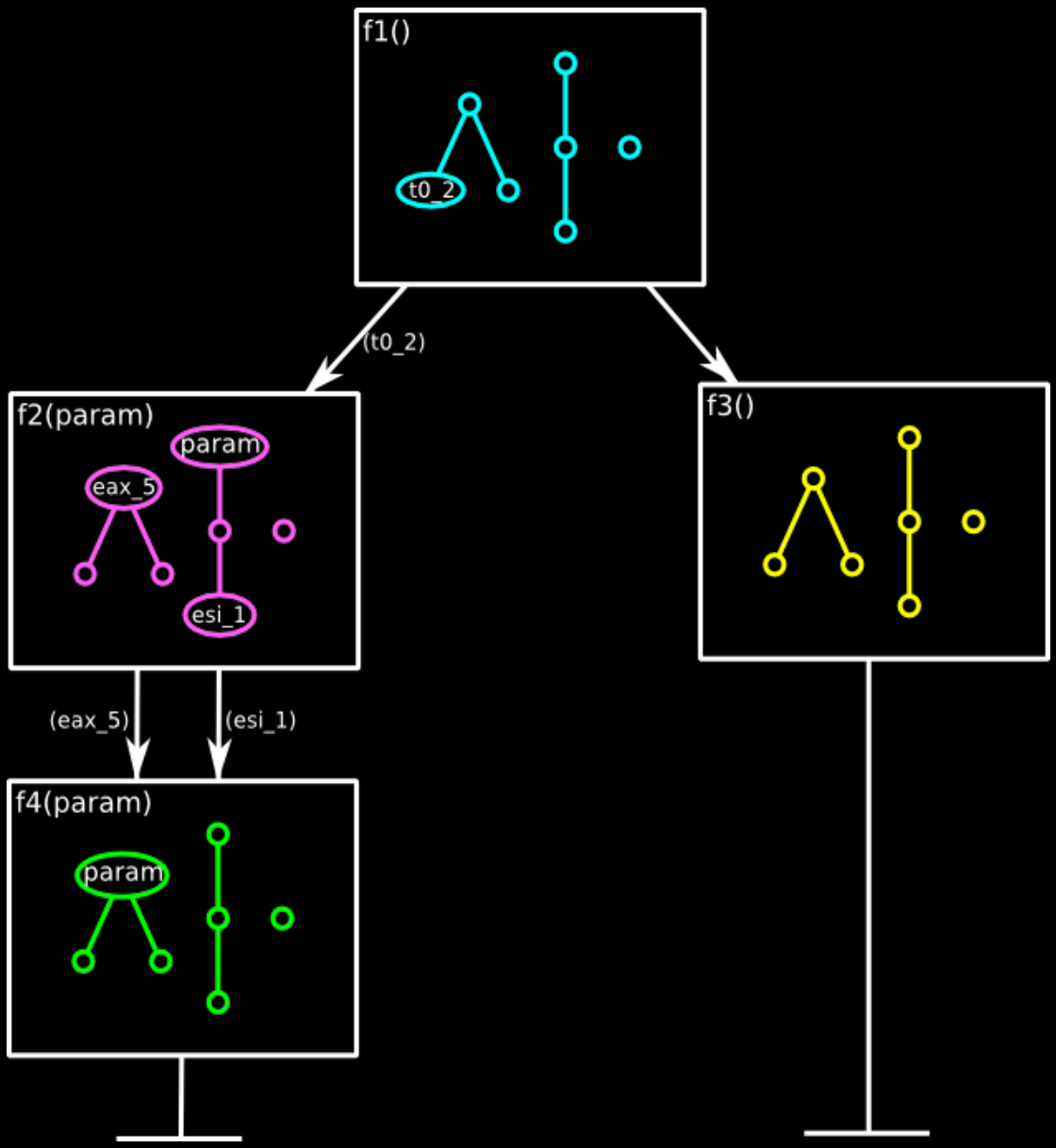
Interprocedural Analysis



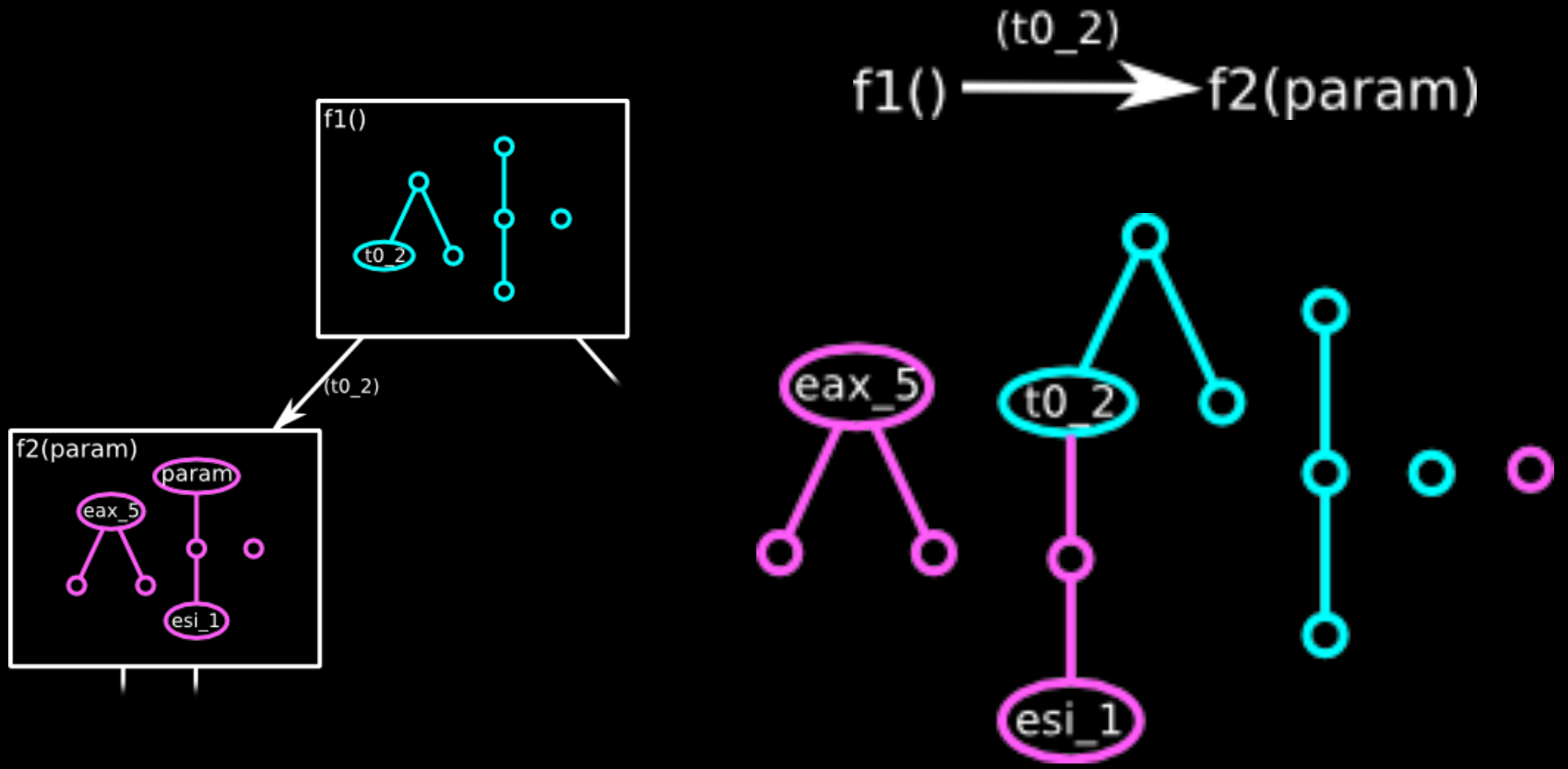
Algorithm

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

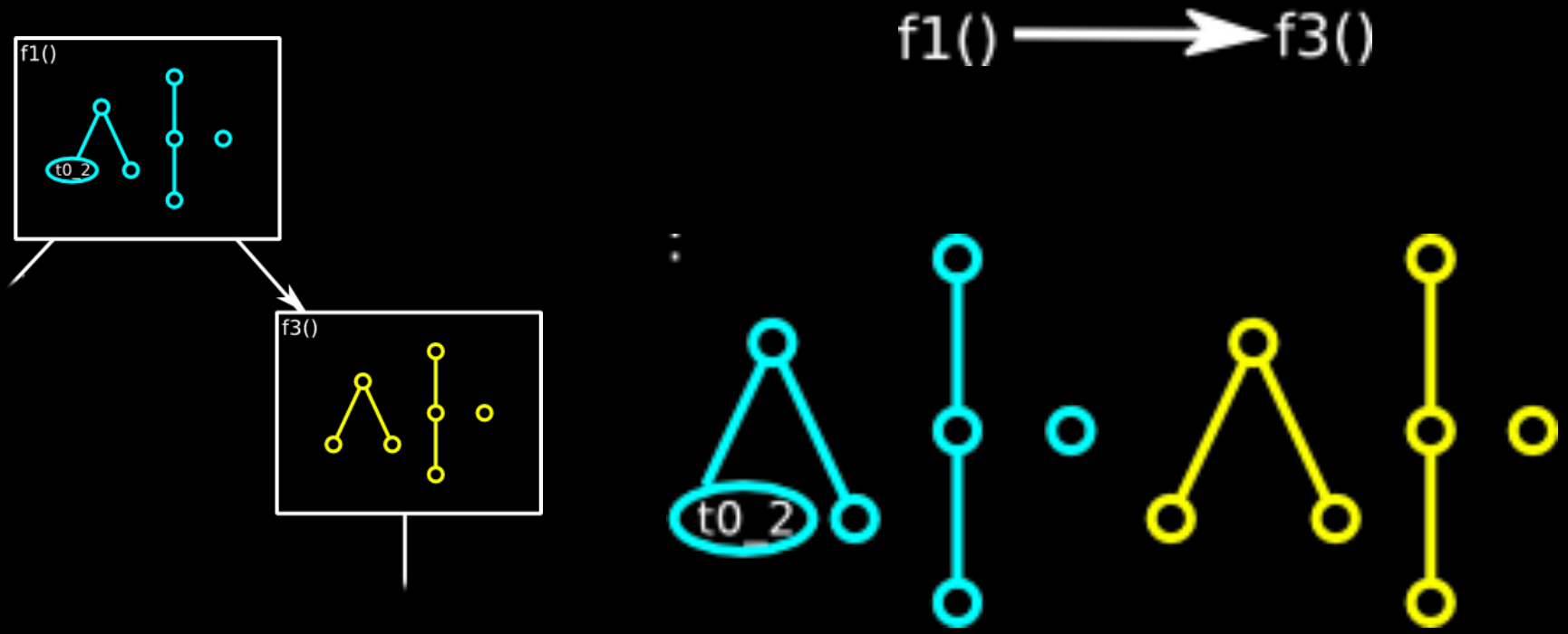
Procedure Call Graph



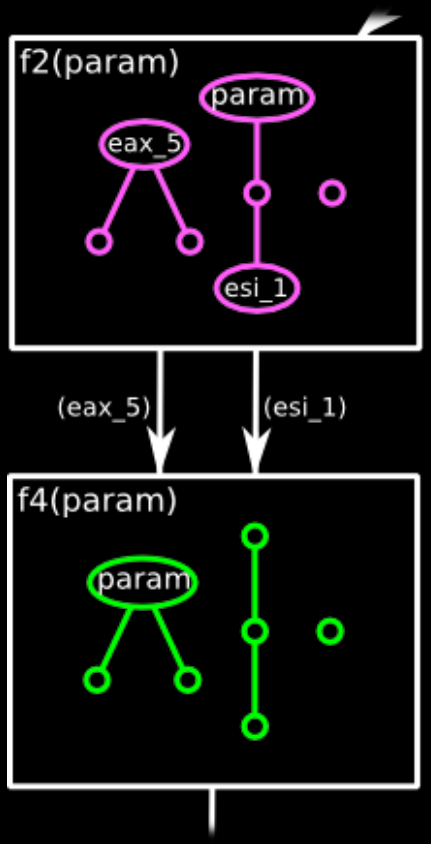
Transformations

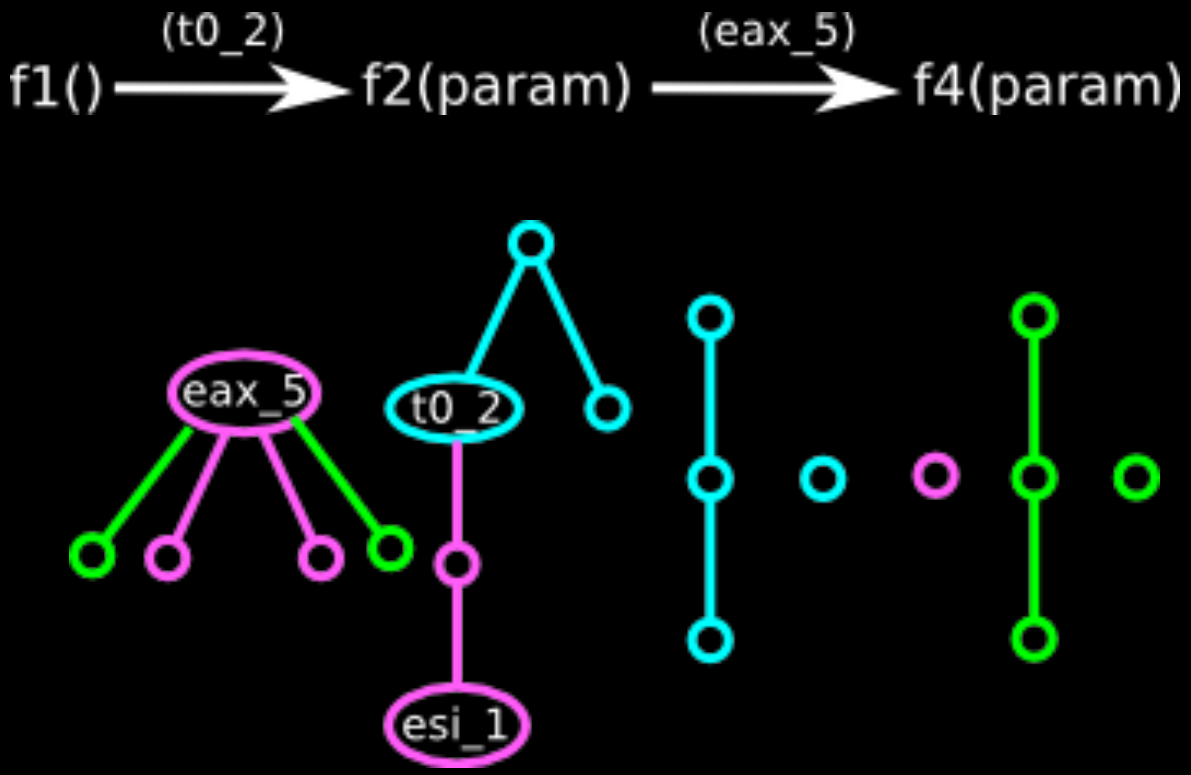


Transformations

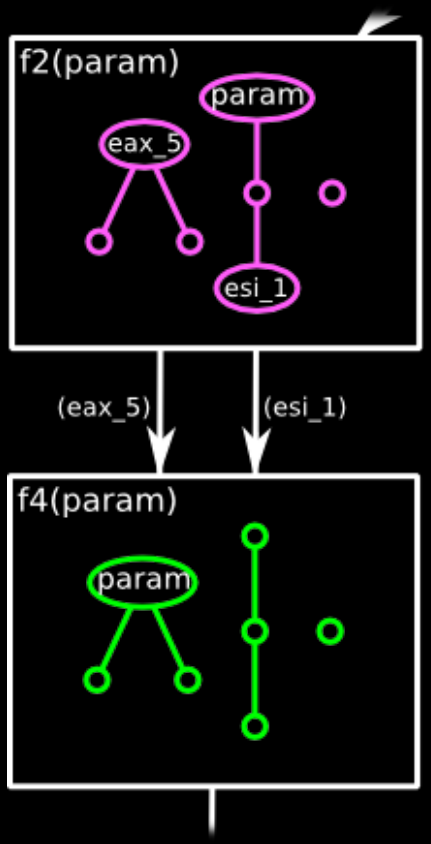


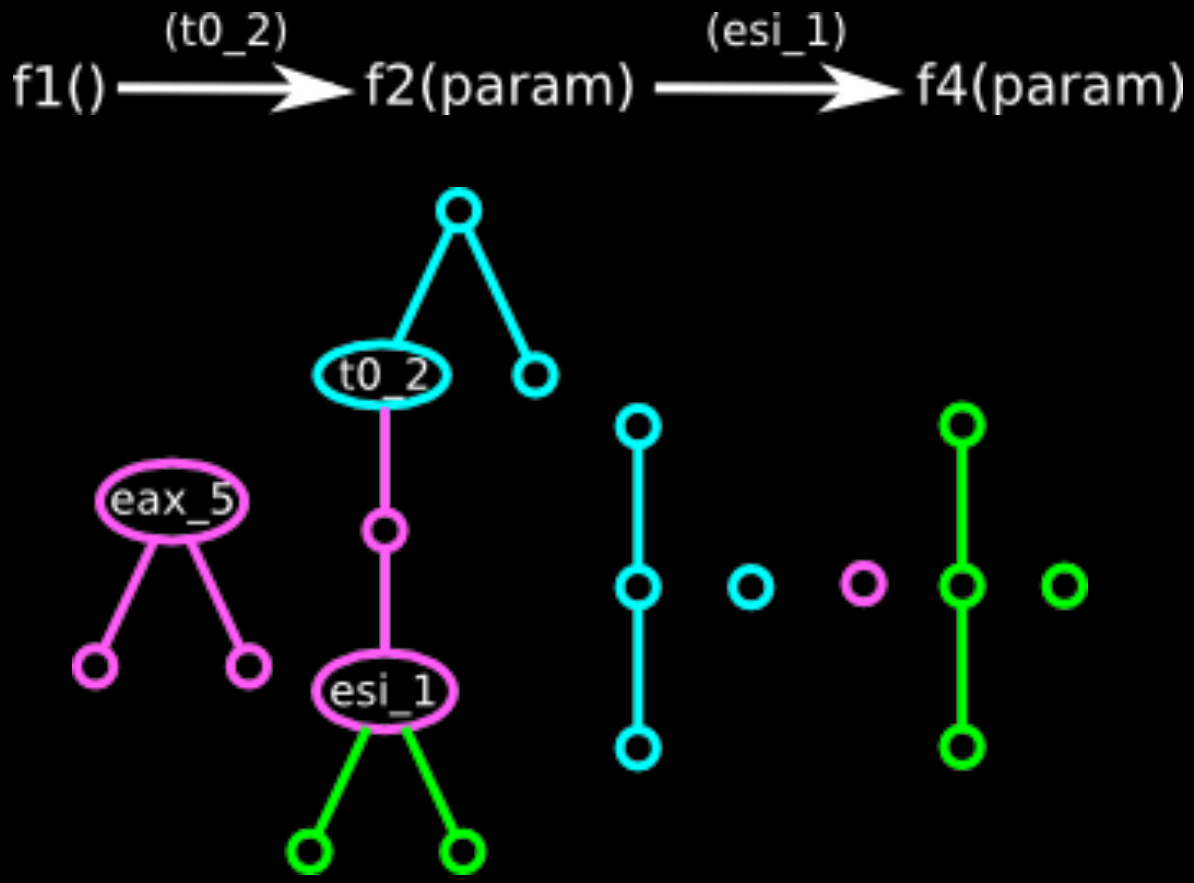
Transformations

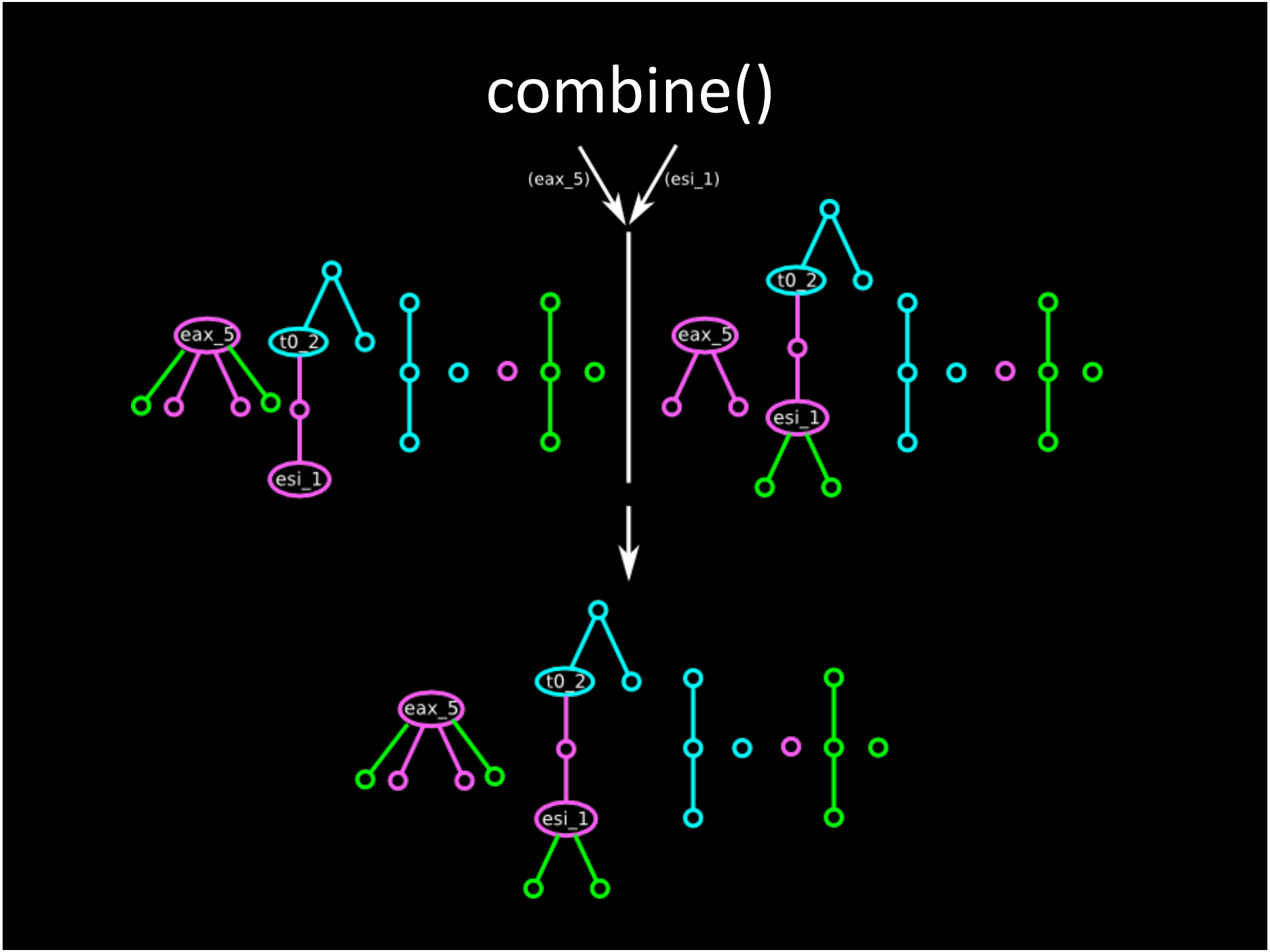




Transformations

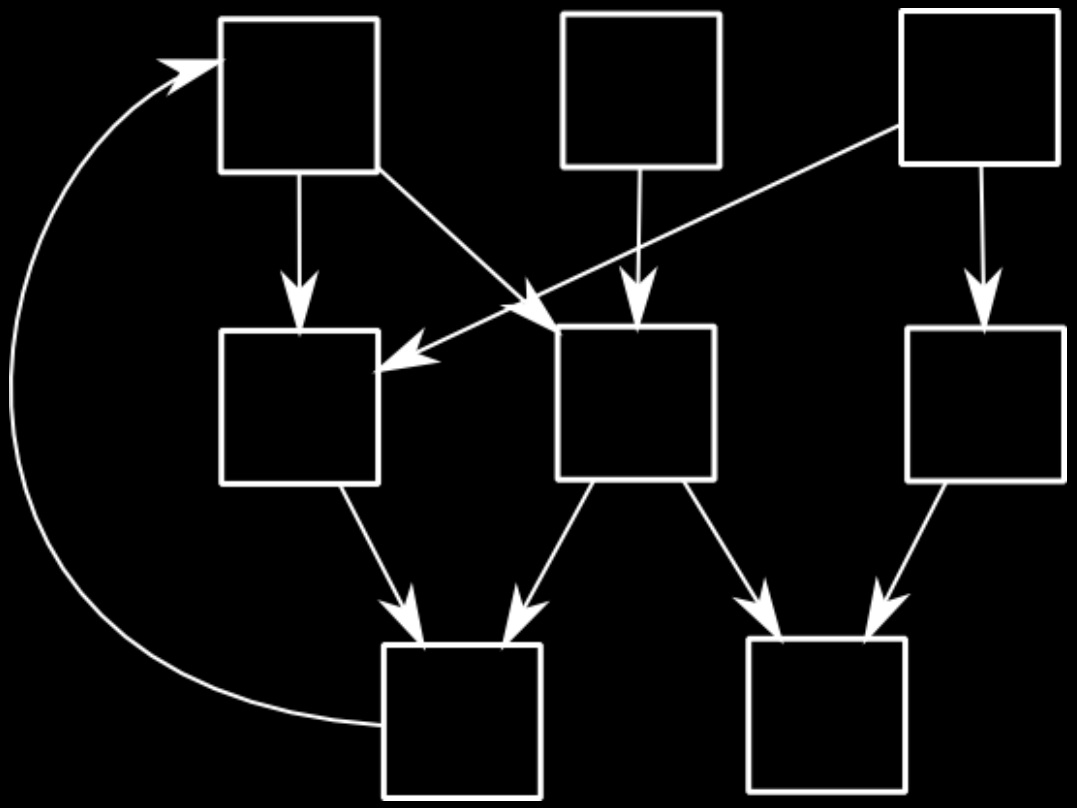




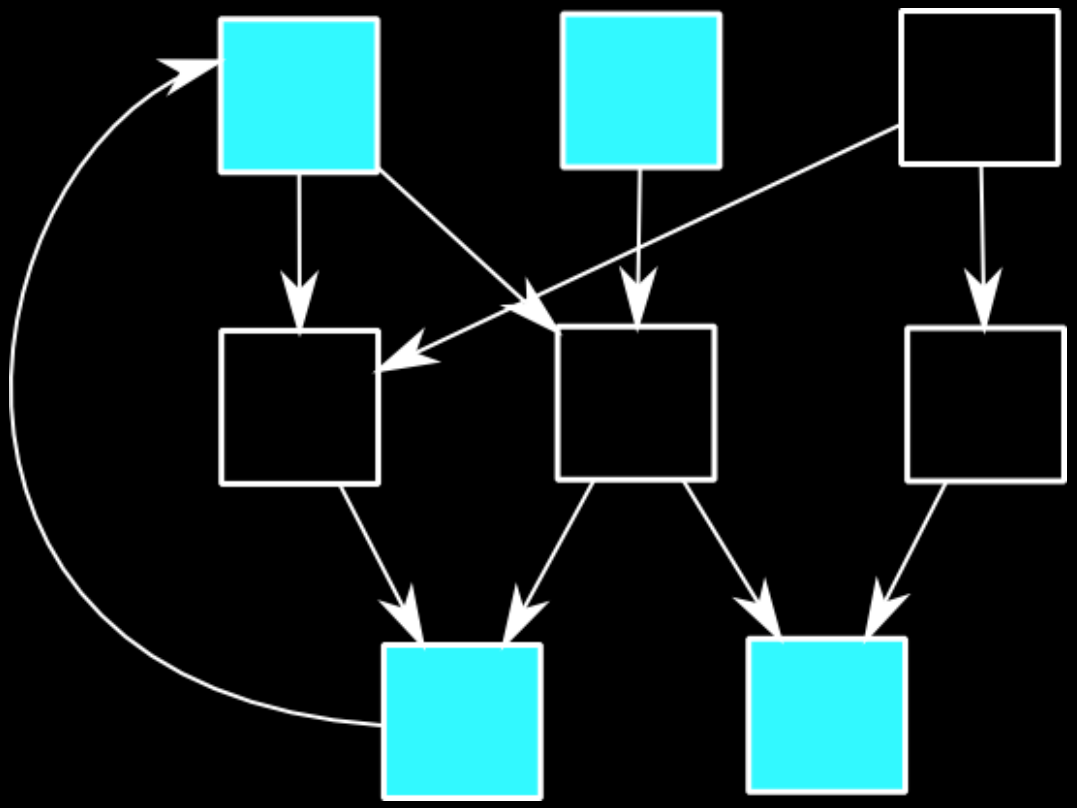

combine()
(eax_5)
(esi_1)
eax_5
t0_2
esi_1
eax_5
t0_2
esi_1
eax_5
t0_2
esi_1

Bug Detection

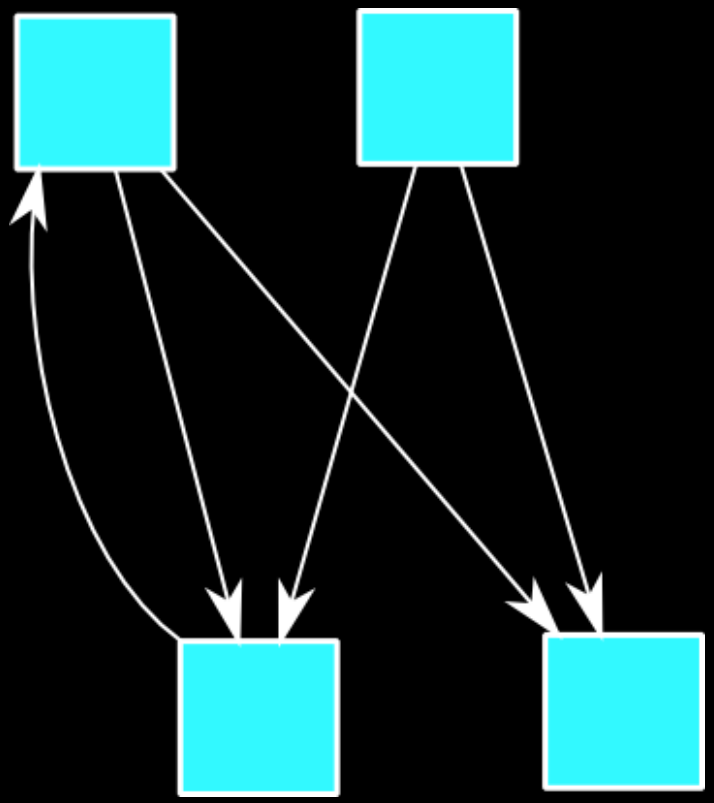
Callgraph pruning



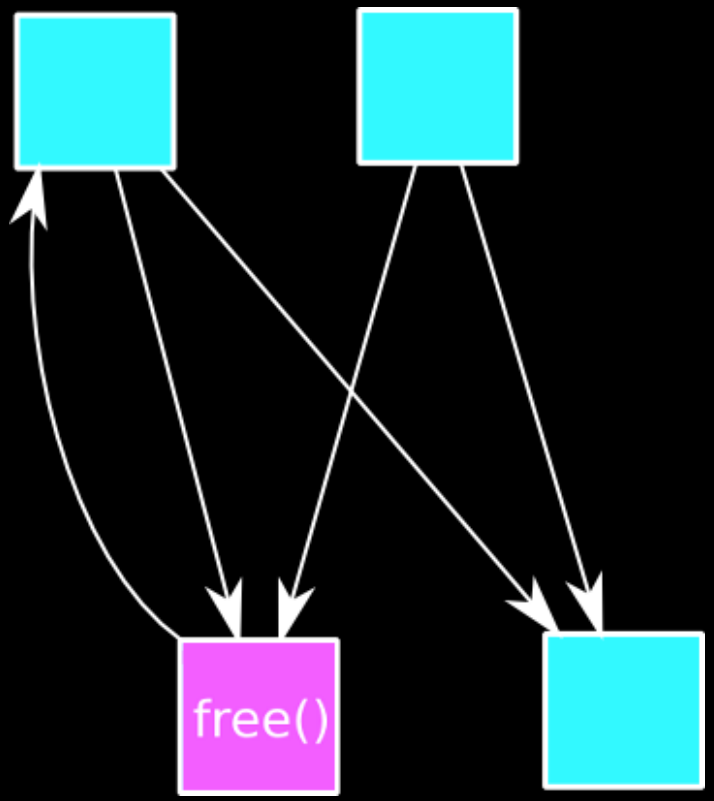
Callgraph pruning



Callgraph pruning



Marking destructor() calls

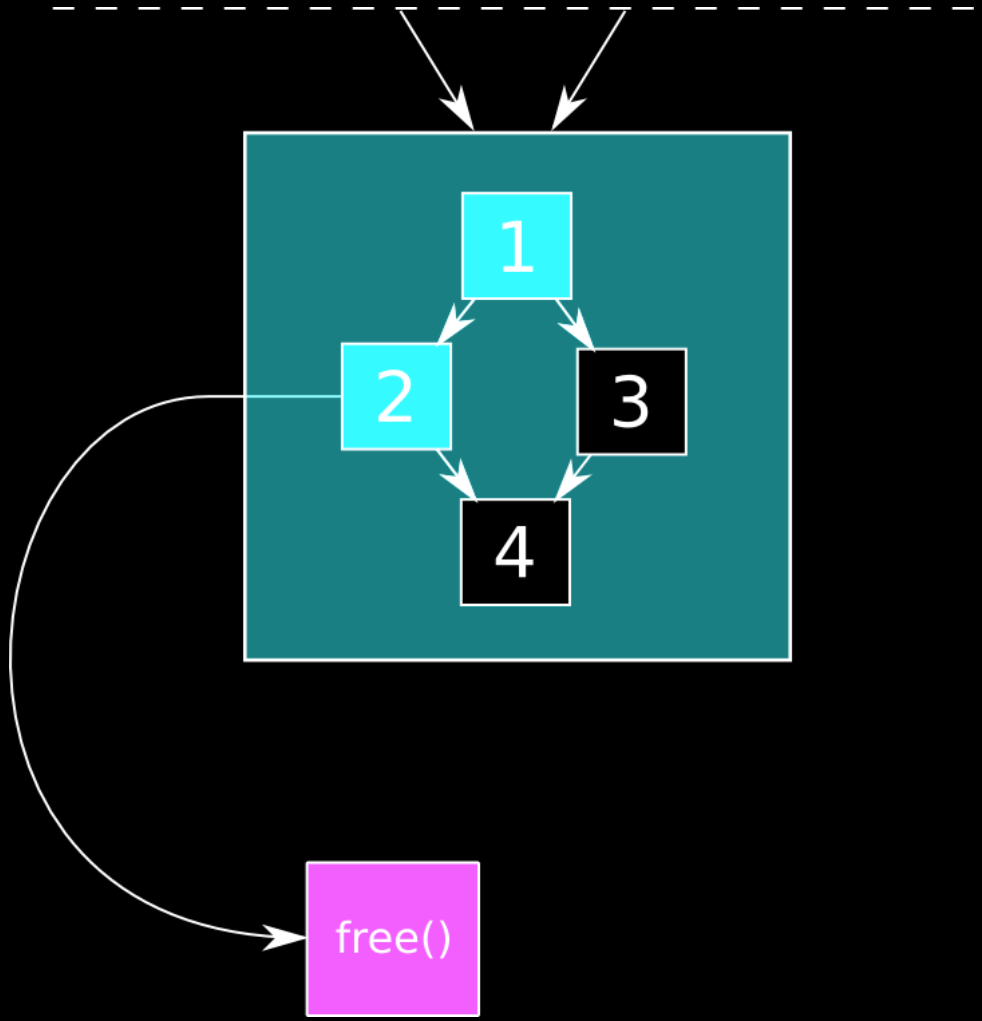


Algorithm

- 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 $B \in dom(X) \Rightarrow$ v is a stale pointer
  - if $B \notin dom(X) \wedge B \in succ(X) \Rightarrow$ v may be a stale pointer
  - if $X \notin dom(B) \wedge X \in succ(B) \Rightarrow$ v may cause memory leak
  - if $X \notin dom(B) \wedge X \notin succ(B) \Rightarrow$ v causes memory leak
- Iterate substituting:
  - F with each of its callers
  - X with a basic block that calls F

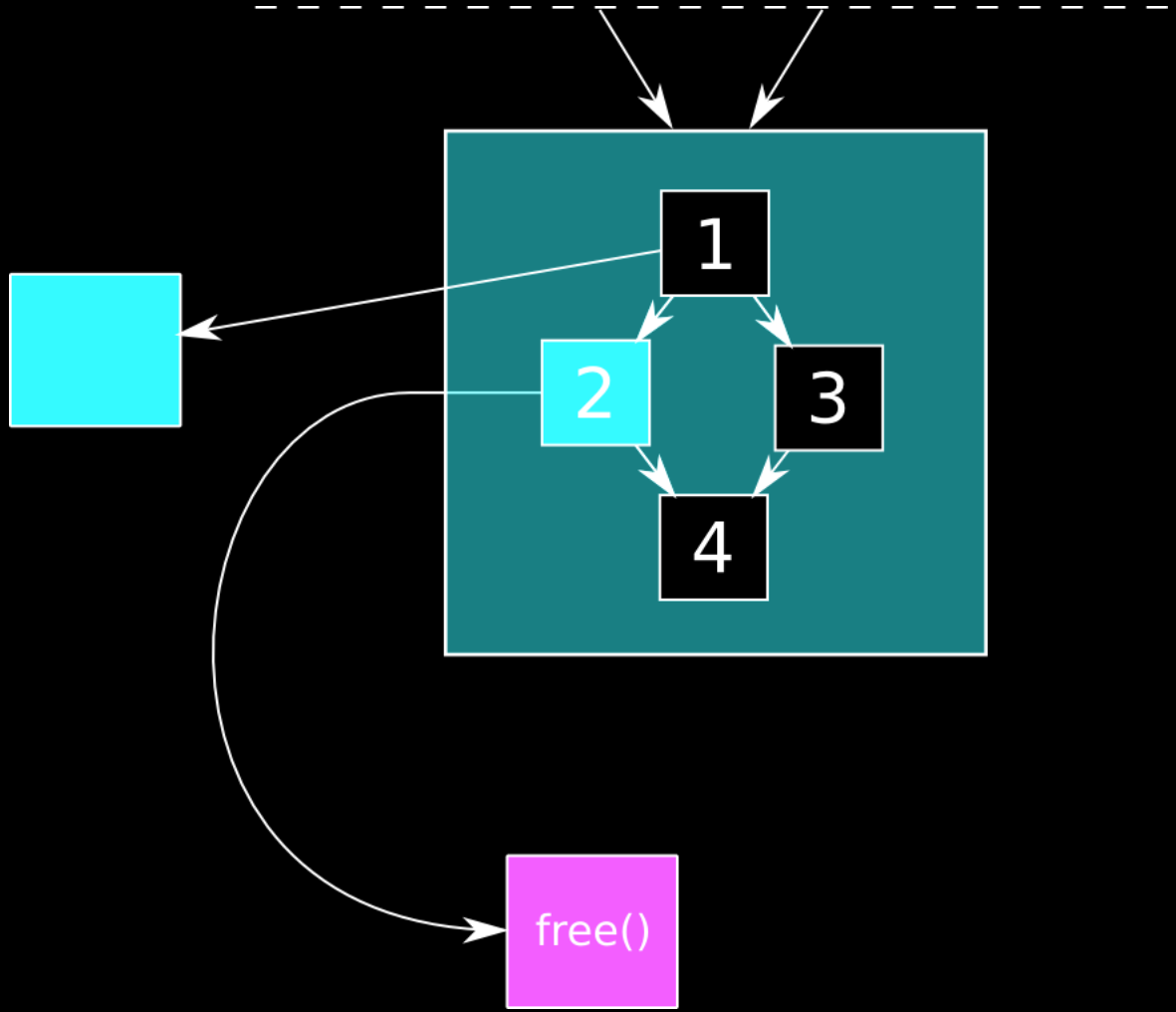
Example

No bugs



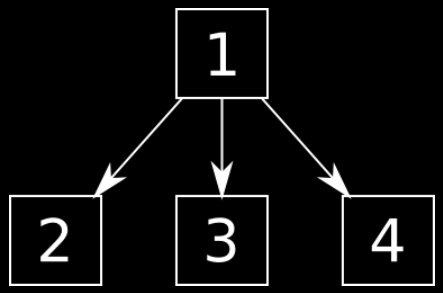
Example

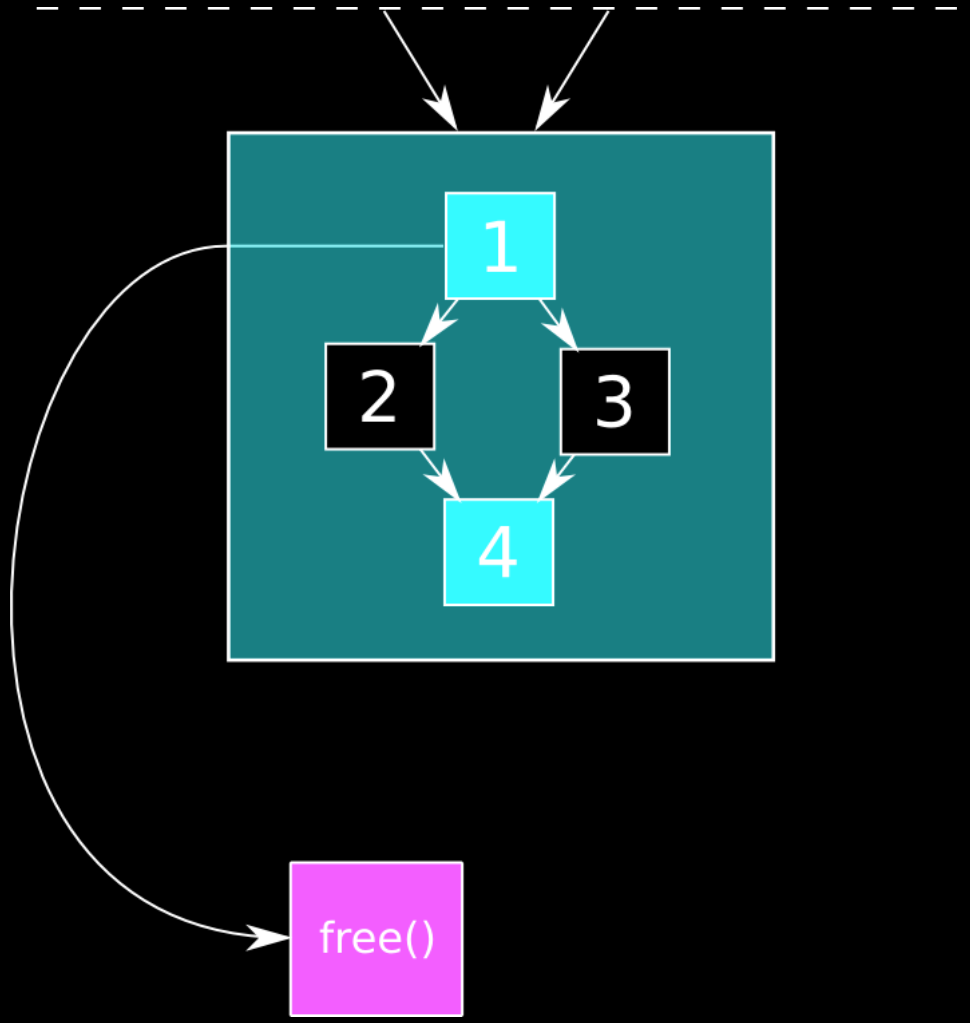
No bugs



Example

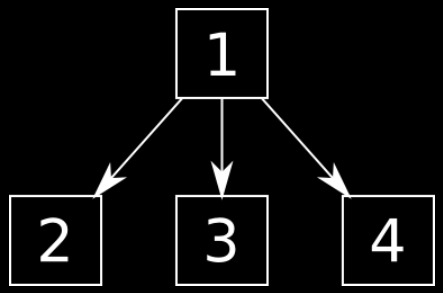
Use after free bug

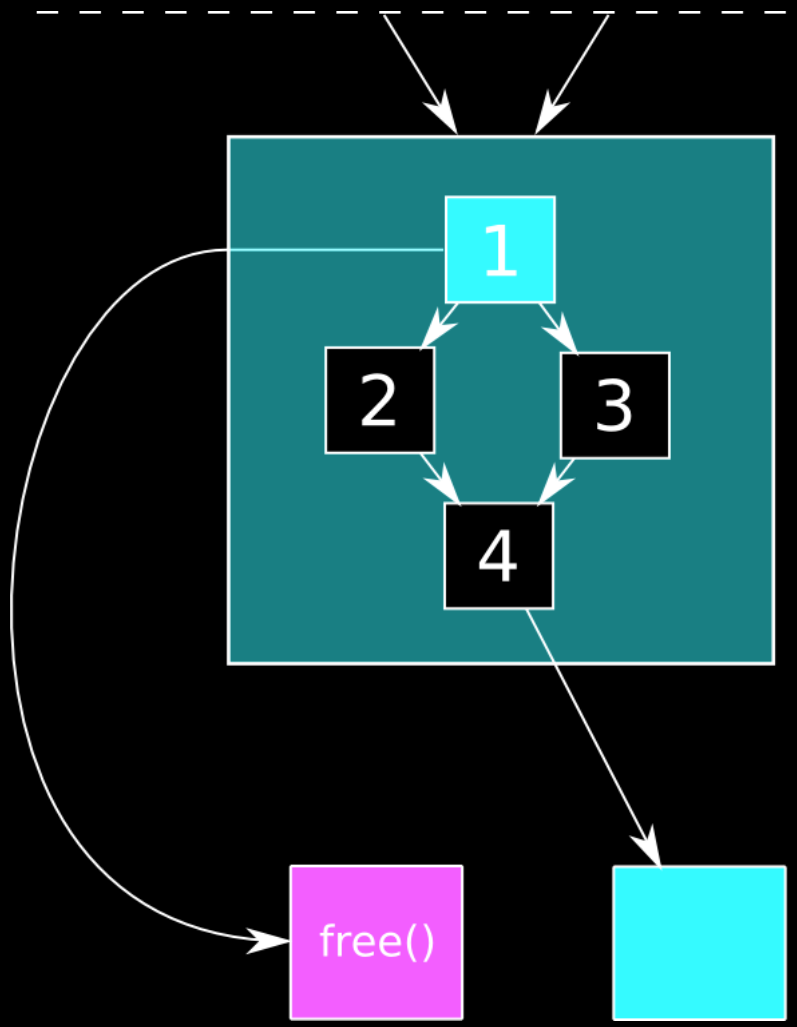




Example

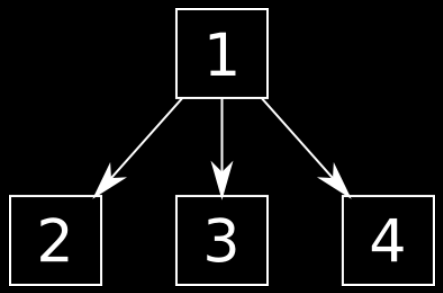
Use after free bug

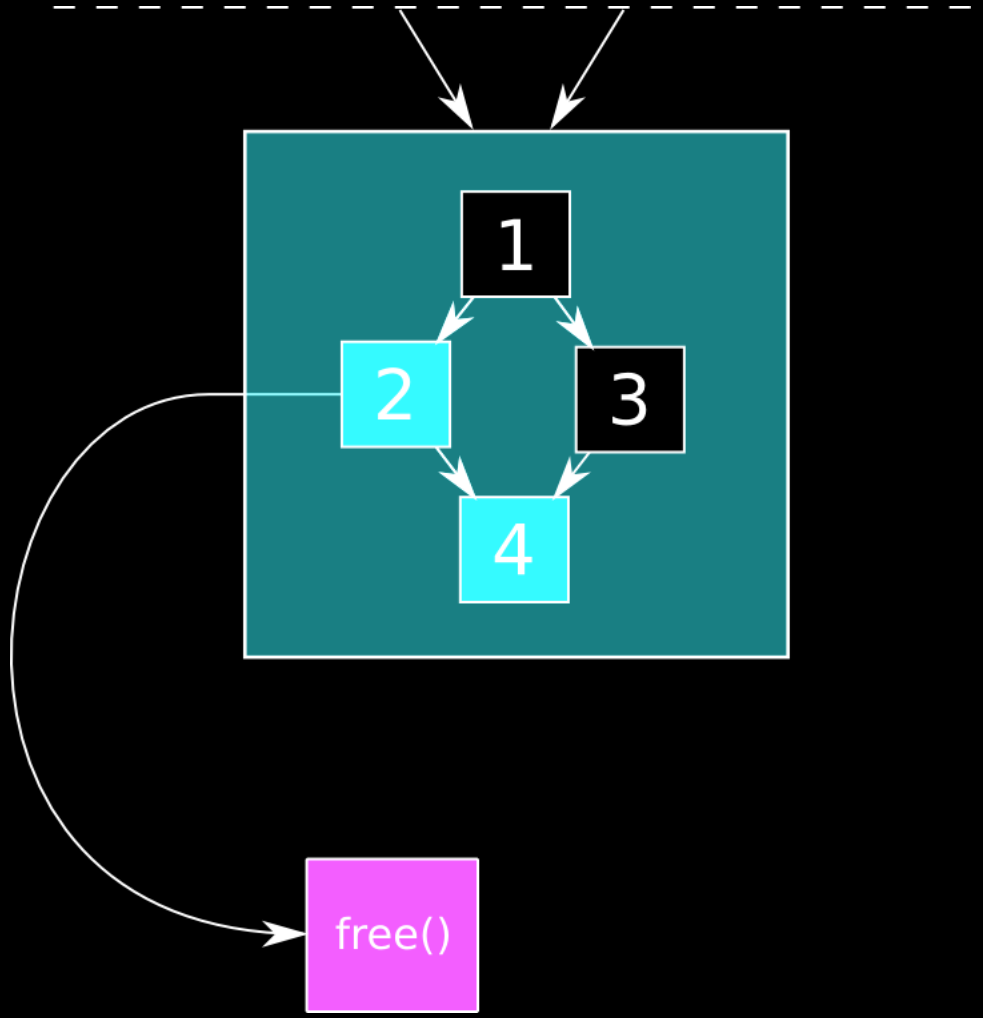




Example

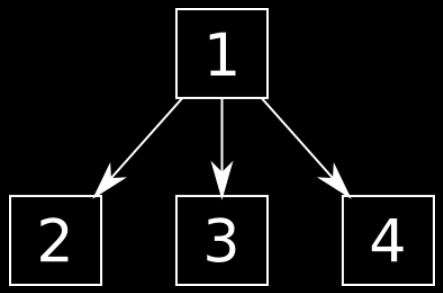
Use after free bug (maybe)

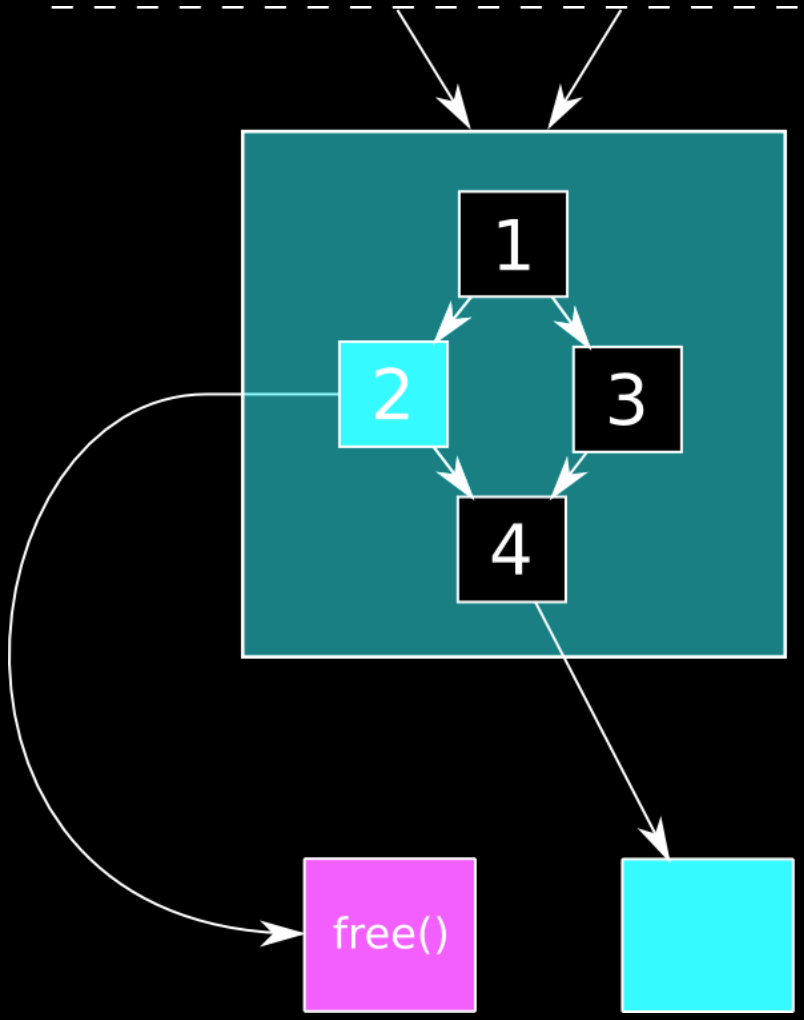




Example

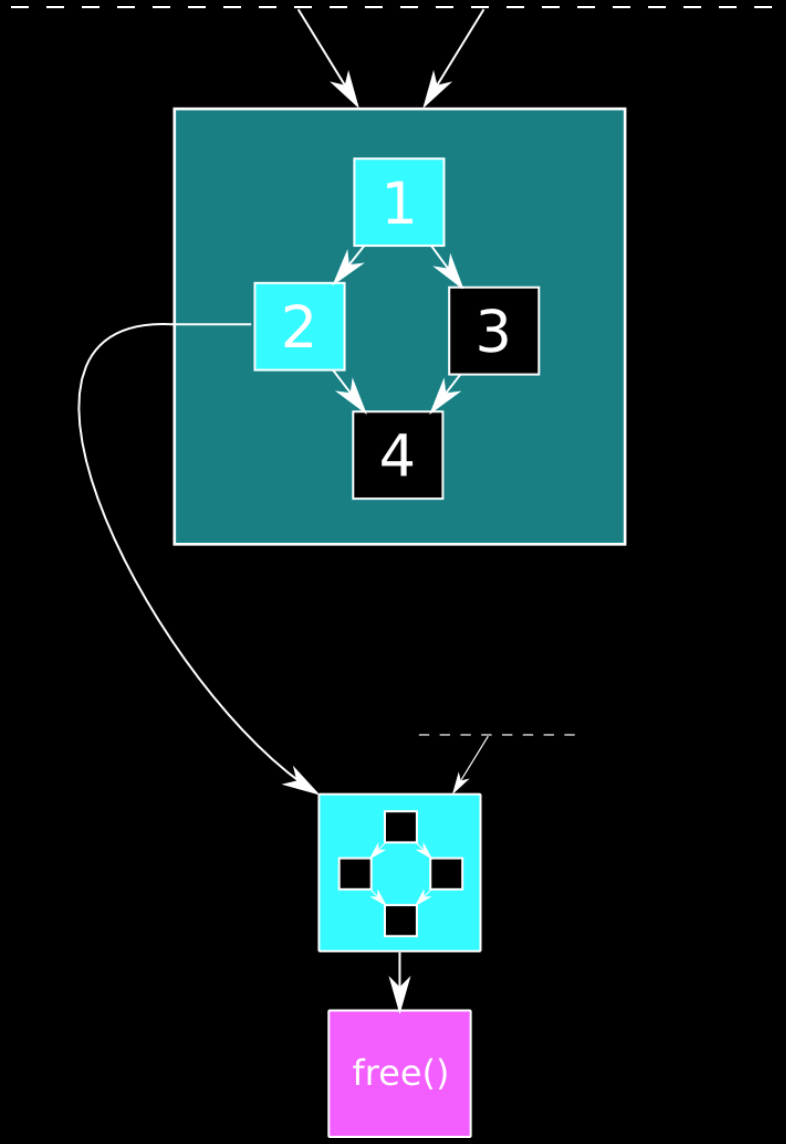
Use after free bug (maybe)





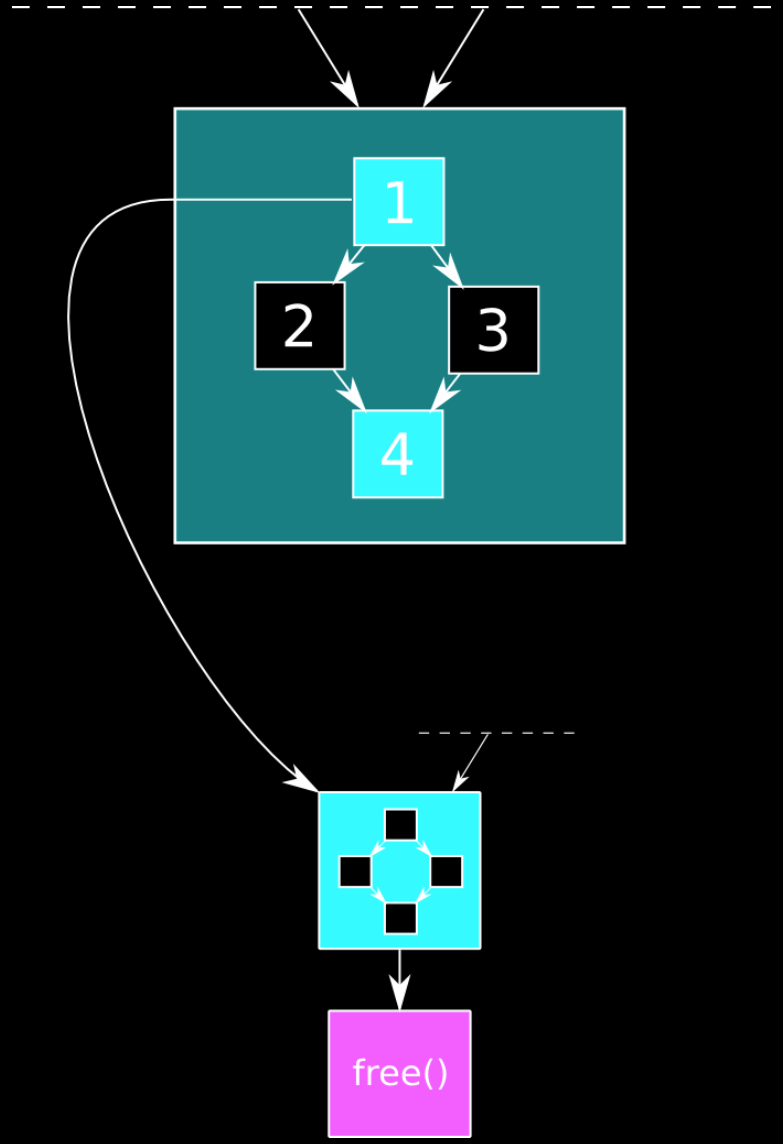
Example

No bugs



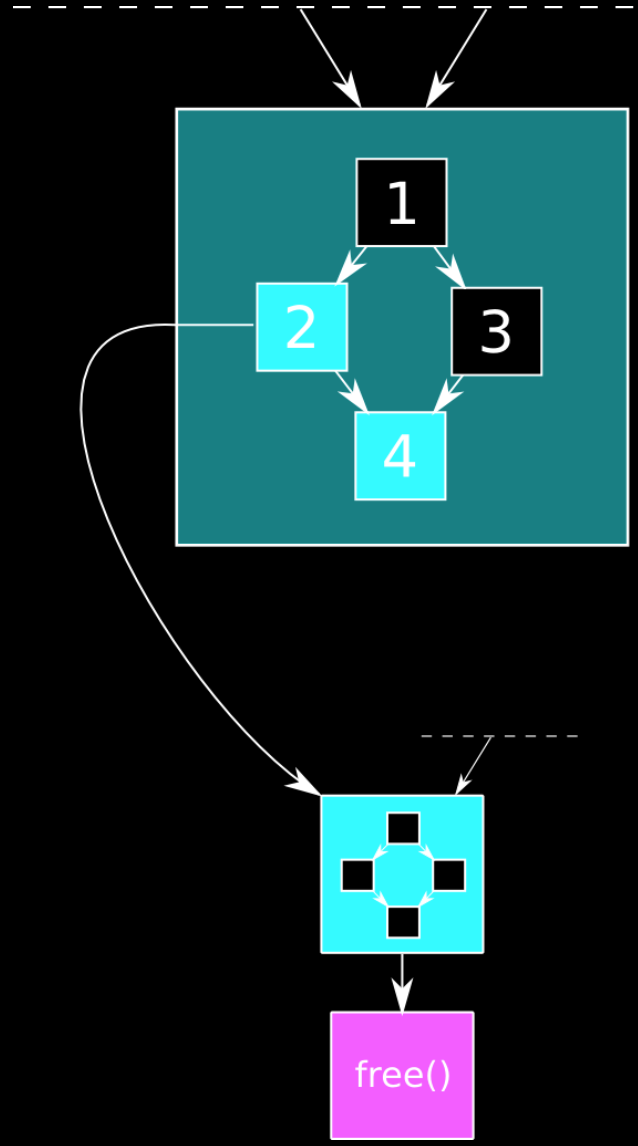
Example

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