

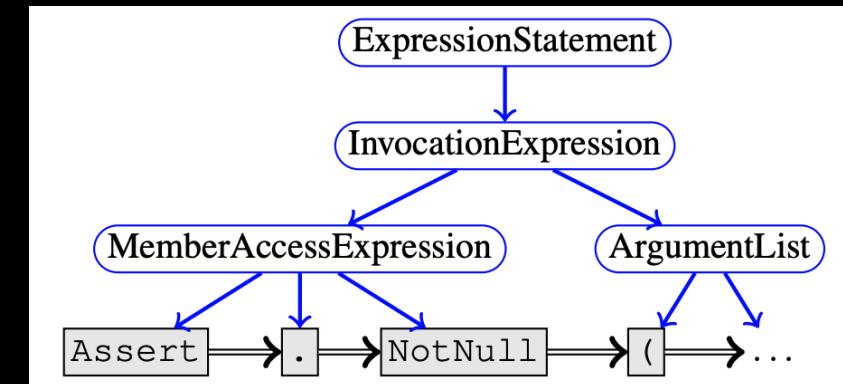
Neural Reverse Engineering of Stripped Binaries using Augmented Control Flow Graphs



Yaniv David, Uri Alon, Eran Yahav
Technion, Israel

Great Progress Using ML on Source Code

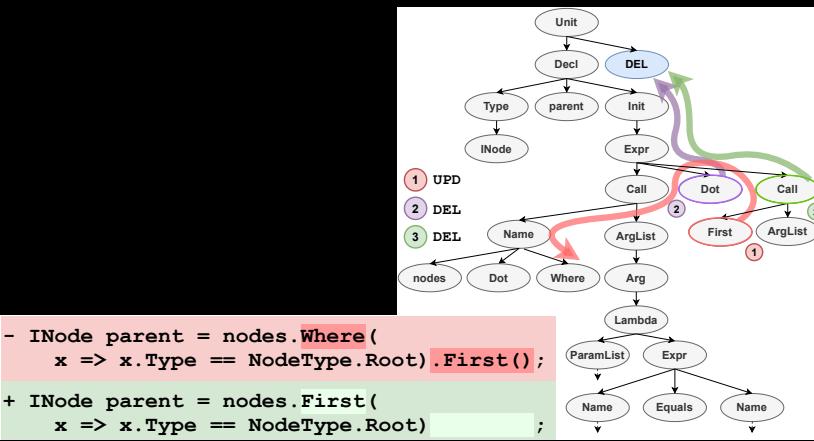
```
public boolean containsIgnoreCase(Set<String> set, String value) {  
    for (String entry : set) {  
        if (entry.equalsIgnoreCase(value))  
            return true;  
    }  
    return false;  
}  
  
contains ① ignore ② case ③
```



[“code2seq”, Alon et al., ICLR’2019]

```
void updateView() {  
    // ...  
    View v = ViewUtil.findView(name);  
    if (v == null) {  
        return;  
    }  
    data.send(((UpdatableView) v)  
        .getContentMgt(), "UPDATE");  
    // ...  
}
```

[Program graphs, Allamanis et al., ICLR’2018]



[“Getafix”, Bader et al., OOPSLA’2019]

[Edit Completion, Brody et al., OOPSLA’2020]

Not a Lot of Progress for Binaries

- Why?

Source code in a high-level language

- Programmers follow common **patterns** (sort, merge, ...)
- These patterns employ sequences of the language's **syntactic structures**
- Many **global anchors**: types, class structures, frameworks

```
void f(int[] a) {  
    boolean s = true;  
    for (int i = 0; i < a.length && s; i++) {  
        s = false;  
        for (int j = 0; j < a.length - 1 - i; j++) {  
            if (a[j] > a[j+1]) {  
                int temp = a[j];  
                a[j] = a[j+1];  
                a[j+1] = temp;  
                s = true;  
            }  
        }  
    }  
}
```

Rich, structured, expressive &
filled with hints

Disassembled Binary Code

```
...
mov rsi, rdi
mov rdx, 16
mov rax, [rbp-58h]
mov rdi, rax
mov edx, eax
mov rax, [rbp-4h]
mov rax, [rbp-58h]
mov rdi, rax
mov r8, 4
mov rdx, 10
mov esi, 0
lea rcx, [rbp-88h]
mov rdi, rax
...
...
```

- Machine generated, adhering to hardware specifications & limitations
- Few simple control flow instructions:
 - jump & call (+interrupt)
- Optimization causes: entangled computation flows, context dependent

Long unstructured sequence of low-level operations

Disassembled Binary Code

```
...  
mov rsi, rdi  
mov rdx, 16  
mov rax, [rbp-58h]  
mov rdi, rax  
mov edx, eax  
mov rax, [rbp-4h]  
mov rax, [rbp-58h]  
mov rdi, rax  
mov r8, 4  
mov rdx, 10  
mov esi, 0  
lea rcx, [rbp-88h]  
mov rdi, rax  
...
```

- Machine generated, adhering to hardware specifications & limitations

These put a lot of pressure on the model

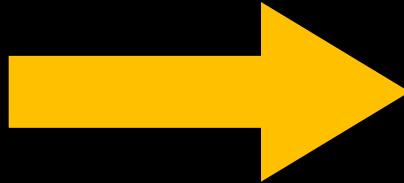
- Optimiz

uses: entangled
computations, loops, context dependent

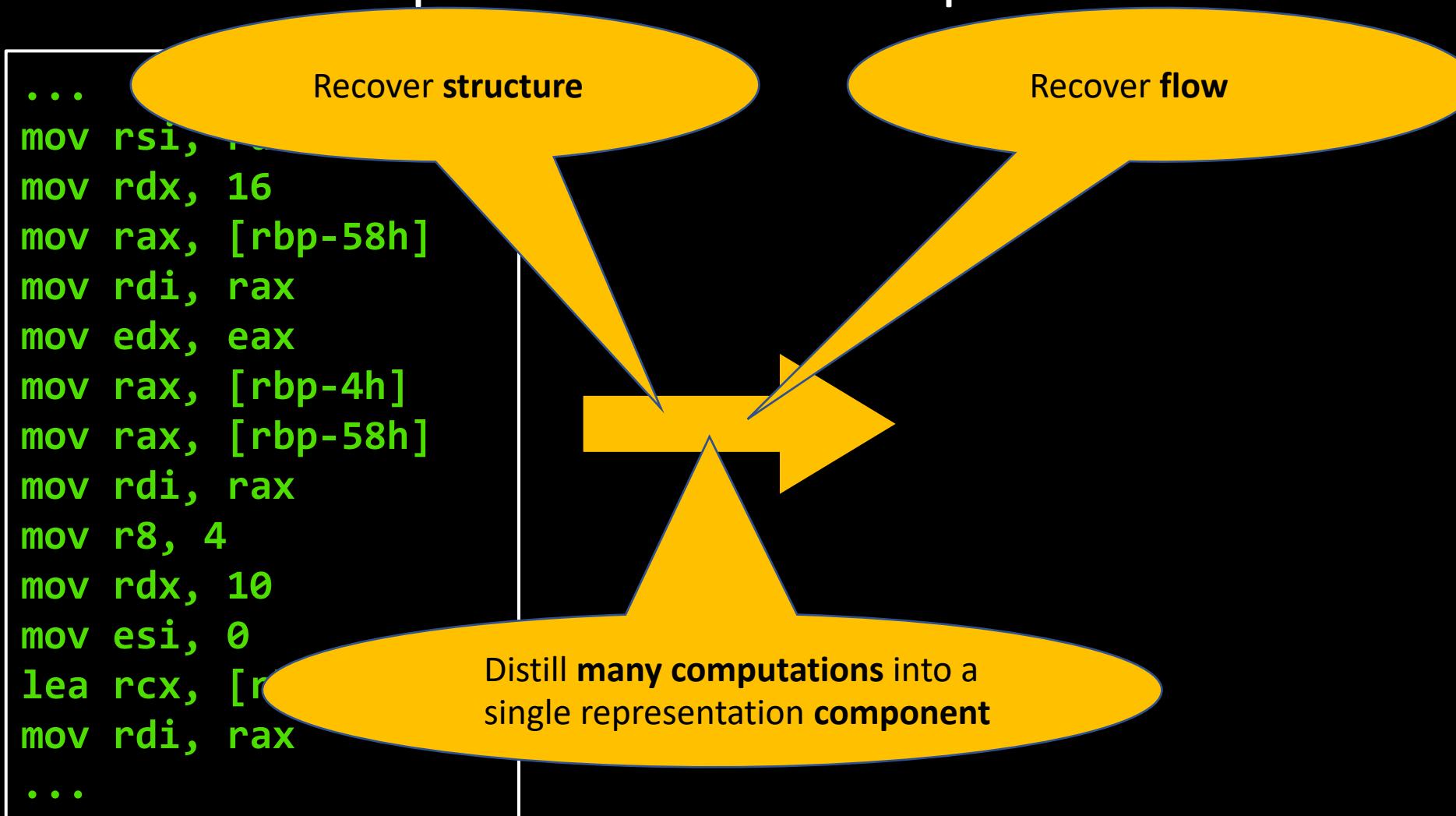
Long unstructured sequence of
low-level operations

Key Idea: Use Binary Program Analysis to Create a Compact & Rich Representation

```
...  
mov rsi, rdi  
mov rdx, 16  
mov rax, [rbp-58h]  
mov rdi, rax  
mov edx, eax  
mov rax, [rbp-4h]  
mov rax, [rbp-58h]  
mov rdi, rax  
mov r8, 4  
mov rdx, 10  
mov esi, 0  
lea rcx, [rbp-88h]  
mov rdi, rax  
...
```

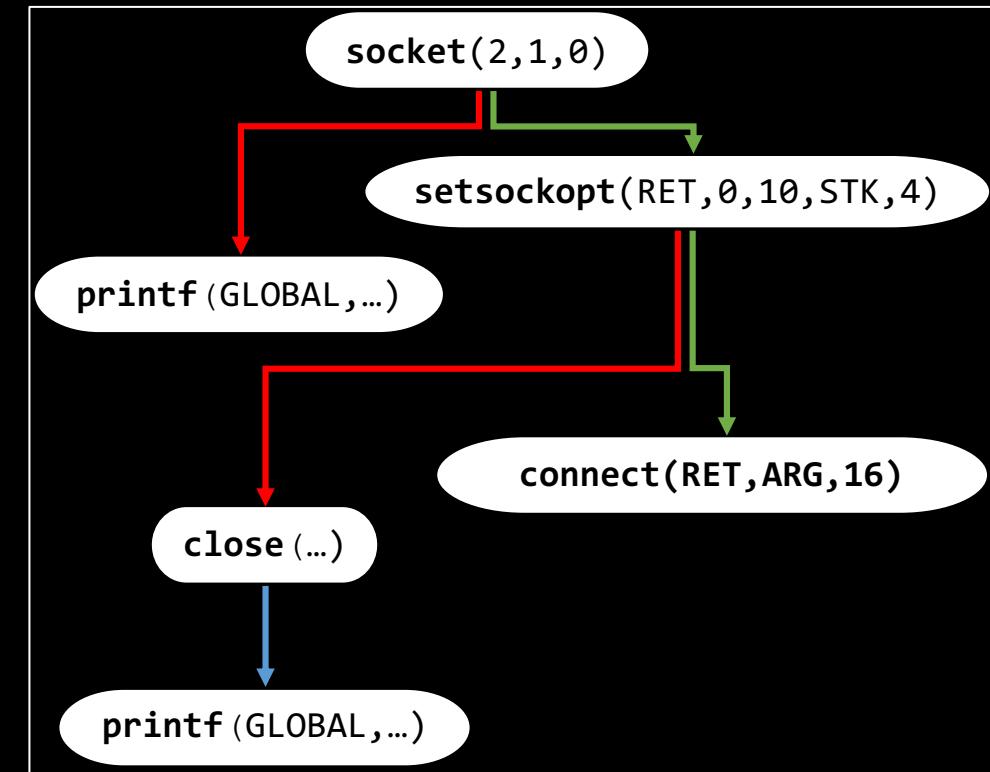
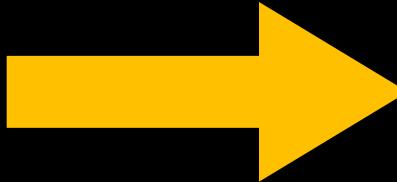


Key Idea: Use Binary Program Analysis to Create a Compact & Rich Representation

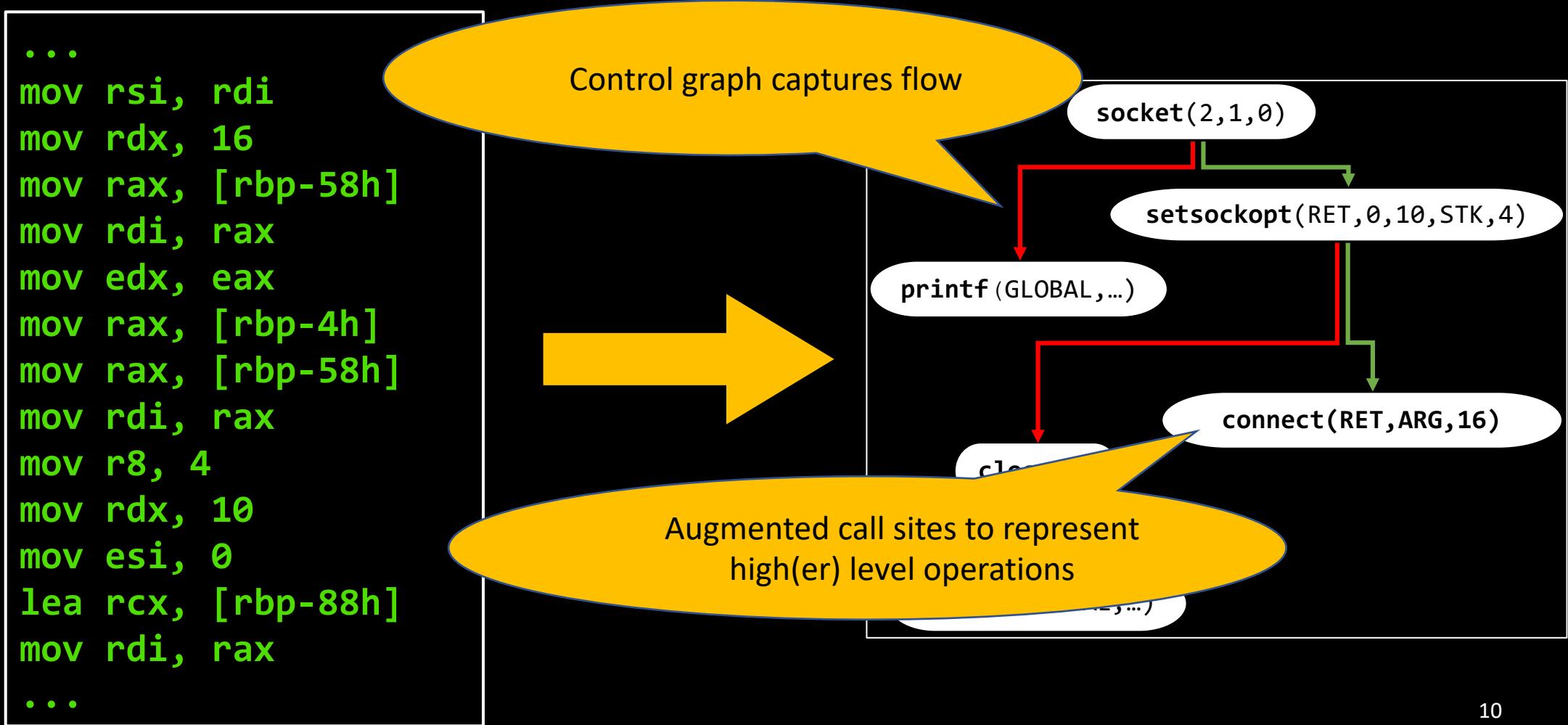


Key Idea: Use Binary Program Analysis to Create a Compact & Rich Representation

```
...  
mov rsi, rdi  
mov rdx, 16  
mov rax, [rbp-58h]  
mov rdi, rax  
mov edx, eax  
mov rax, [rbp-4h]  
mov rax, [rbp-58h]  
mov rdi, rax  
mov r8, 4  
mov rdx, 10  
mov esi, 0  
lea rcx, [rbp-88h]  
mov rdi, rax  
...
```



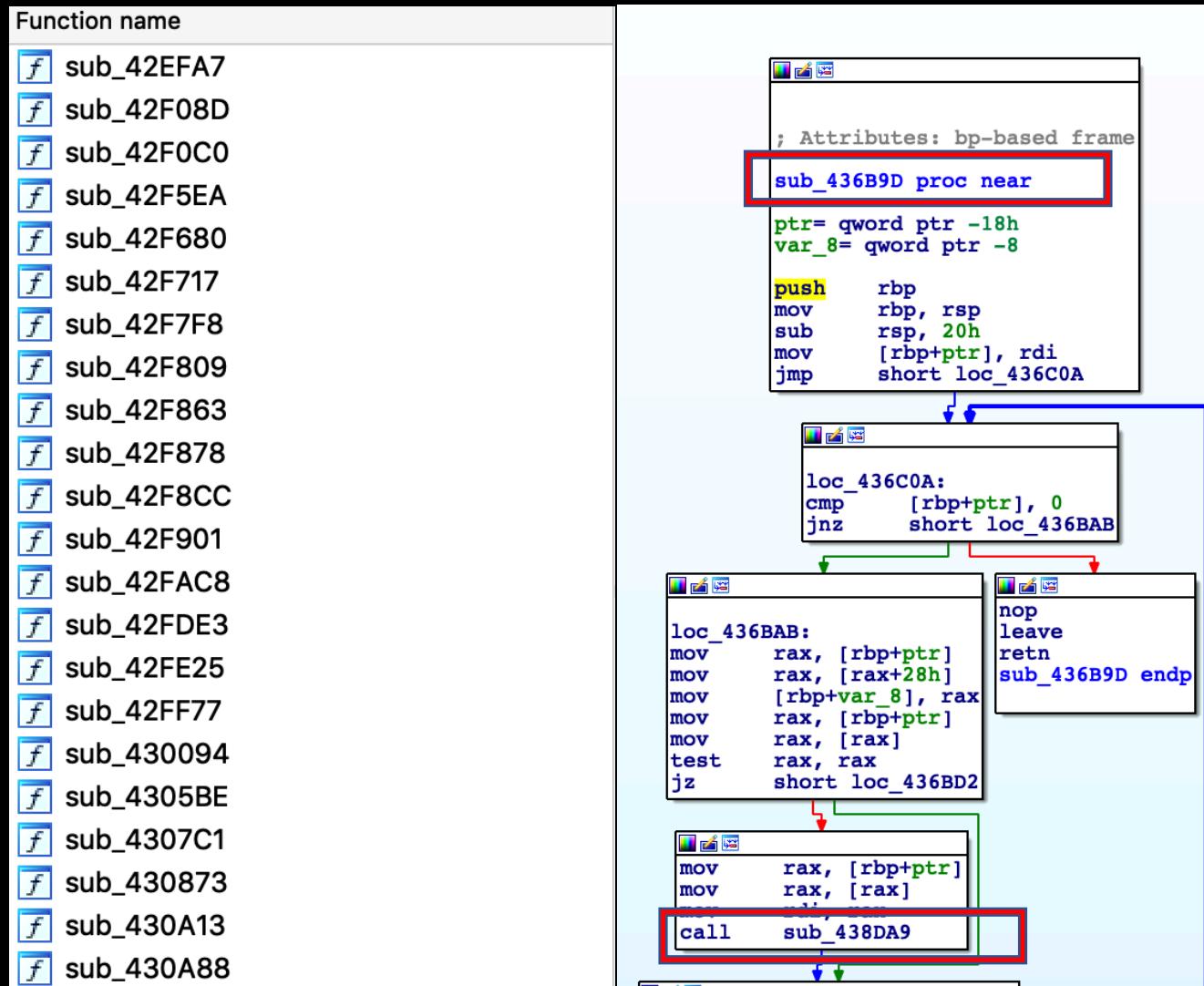
Key Idea: Use Binary Program Analysis to Create a Compact & Rich Representation



Motivating Task

Naming Procedures in Binaries

Helping Reverse Engineers



Helping Reverse Engineers

Function name

- sub_42EFA7
- sub_42F08D
- sub_42F0C0

Where to start?

- sub_42F80
- sub_42F863
- sub_42F878
- sub_42F8CC
- sub_42F901
- sub_42FAC8
- sub_42FDE3
- sub_42FE25
- sub_42FF77
- sub_430094
- sub_4305BE
- sub_4307C1
- sub_430873
- sub_430A13
- sub_430A88

```
; Attributes: bp-based frame
sub_436B9D proc near
ptr= qword ptr -18h
var_8= qword ptr -8

push    rbp
mov     rbp, rsp
sub    rsp, 20h
mov     [rbp+ptr], rdi
jmp     short loc_436C0A

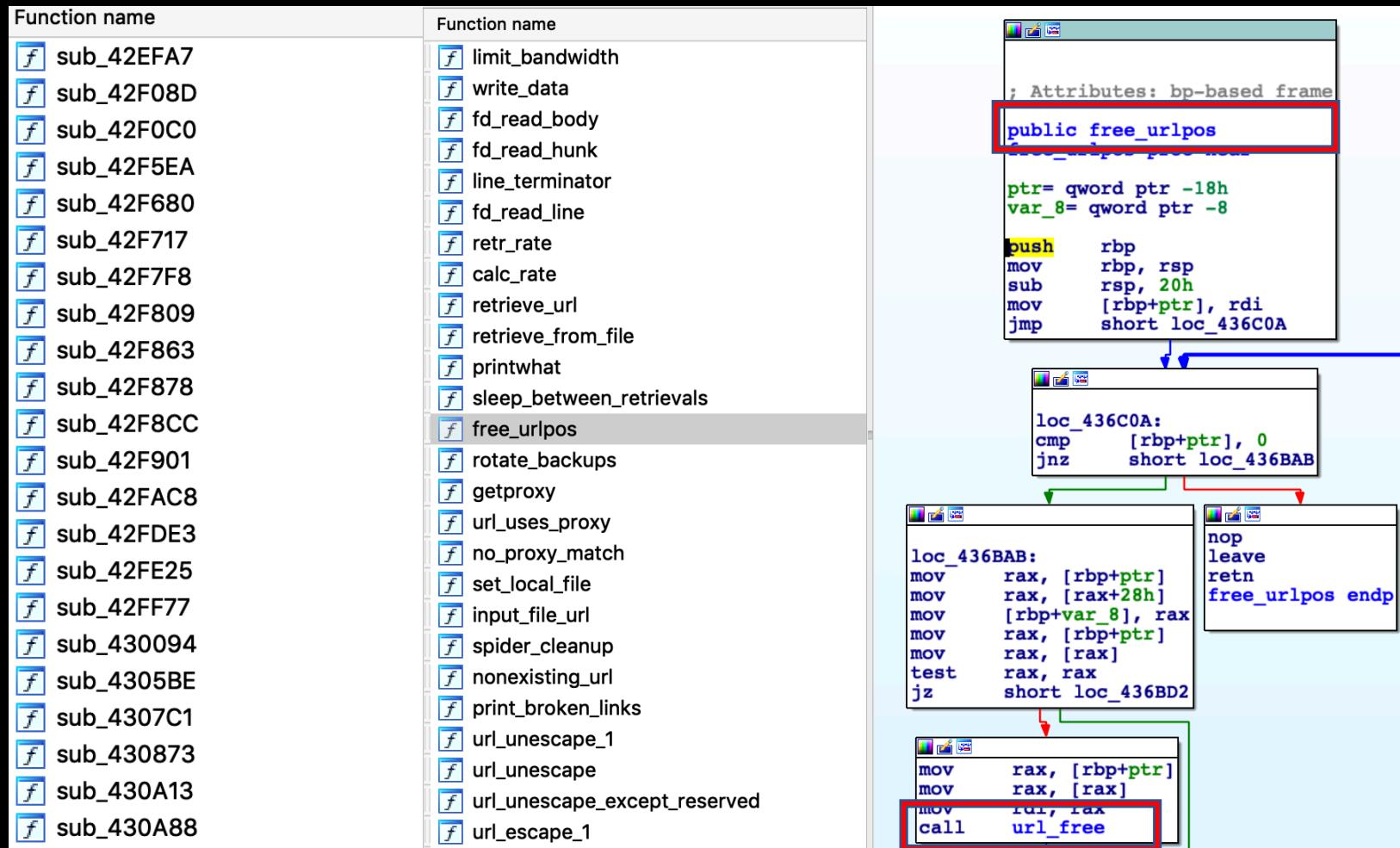
loc_436C0A:
cmp     [rbp+ptr], 0
jnz     short loc_436BAB

loc_436BAB:
mov     rax, [rbp+ptr]
mov     rax, [rax+28h]
mov     [rbp+var_8], rax
mov     rax, [rbp+ptr]
mov     rax, [rax]
test   rax, rax
jz      short loc_436BD2

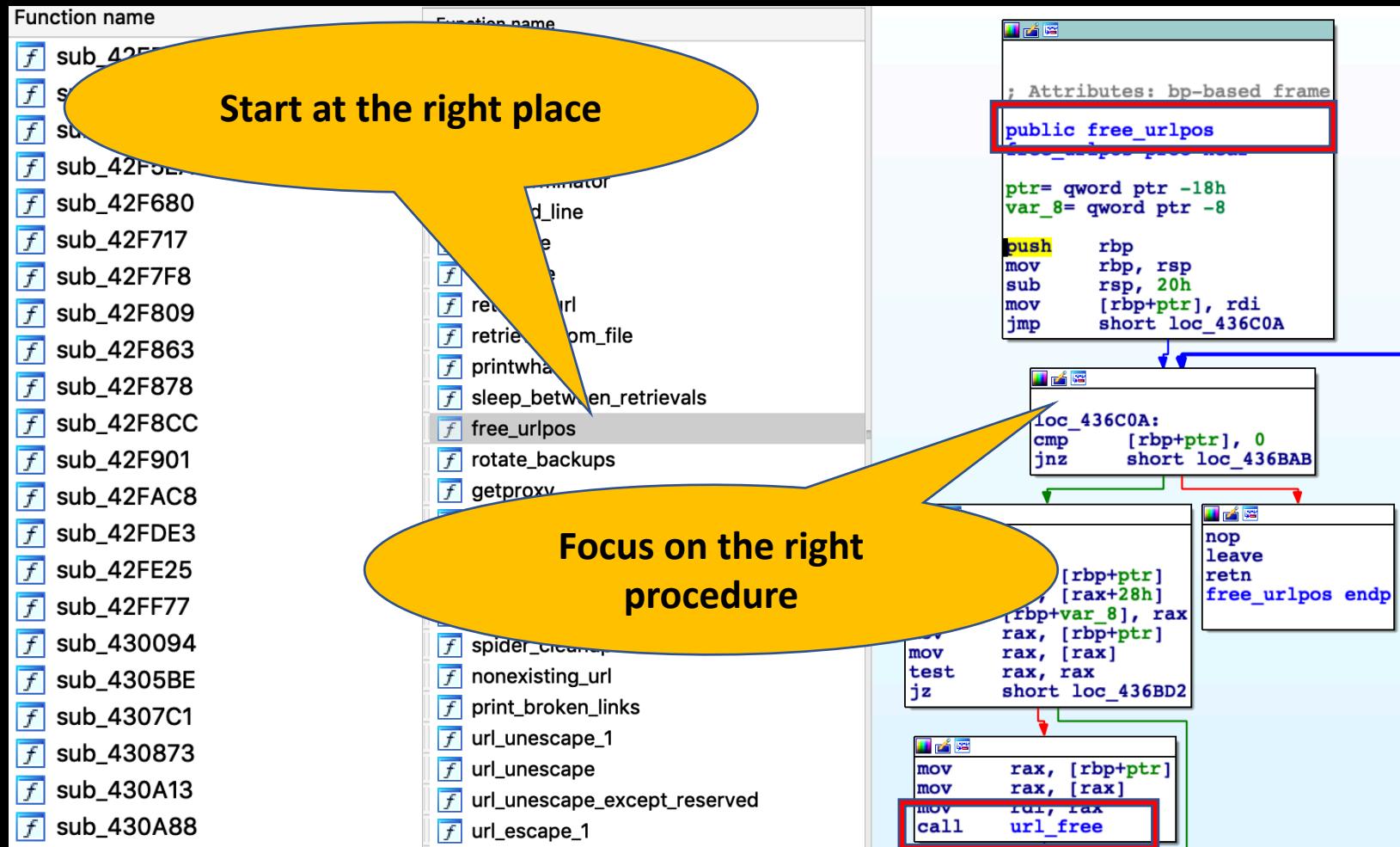
loc_436BD2:
mov     rax, [rbp+ptr]
mov     rax, [rax]
call   sub_438DA9

nop
leave
retn
sub_436B9D endp
```

Un-Stripping Procedure Names

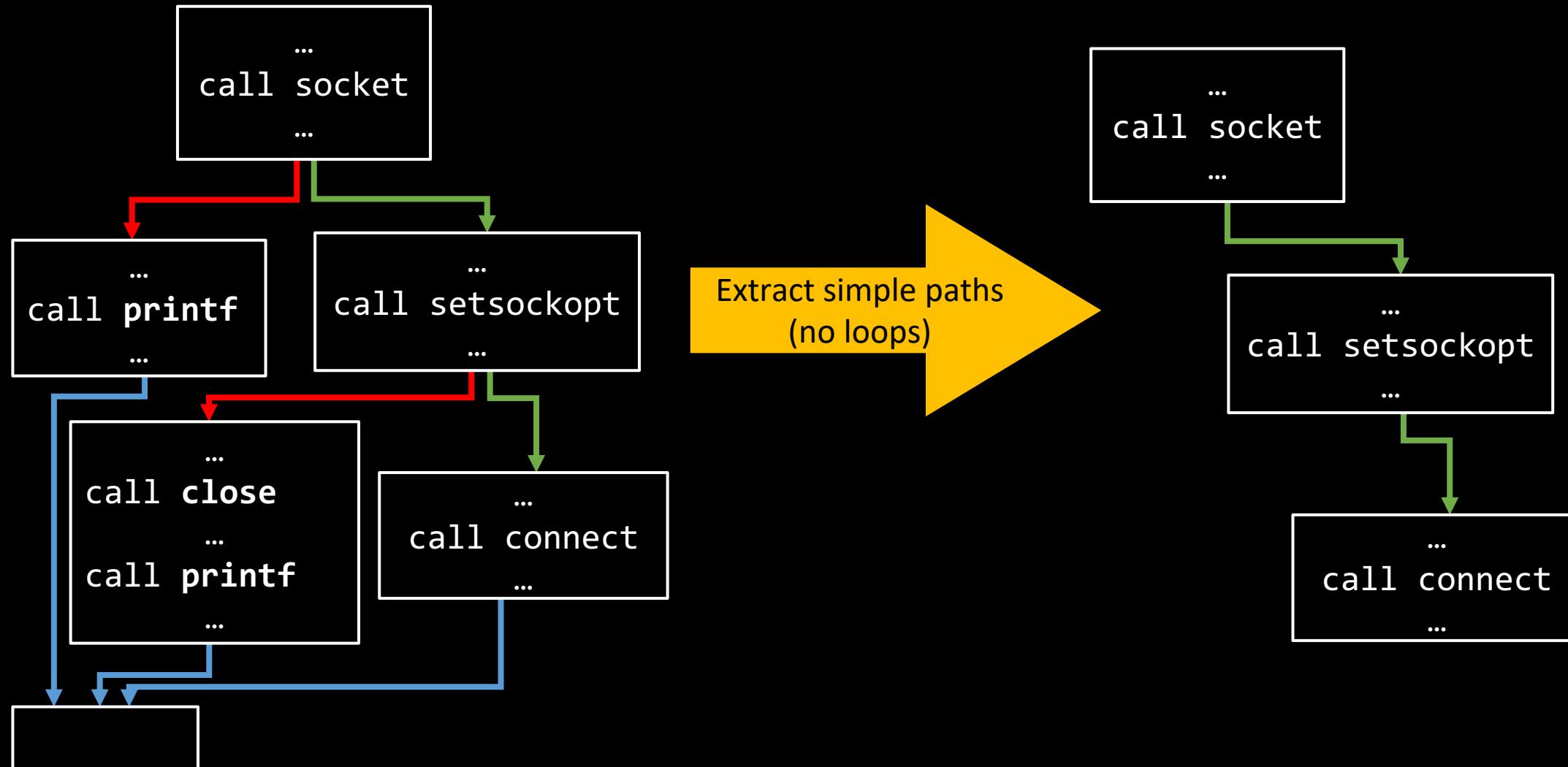


Un-Stripping Procedure Names

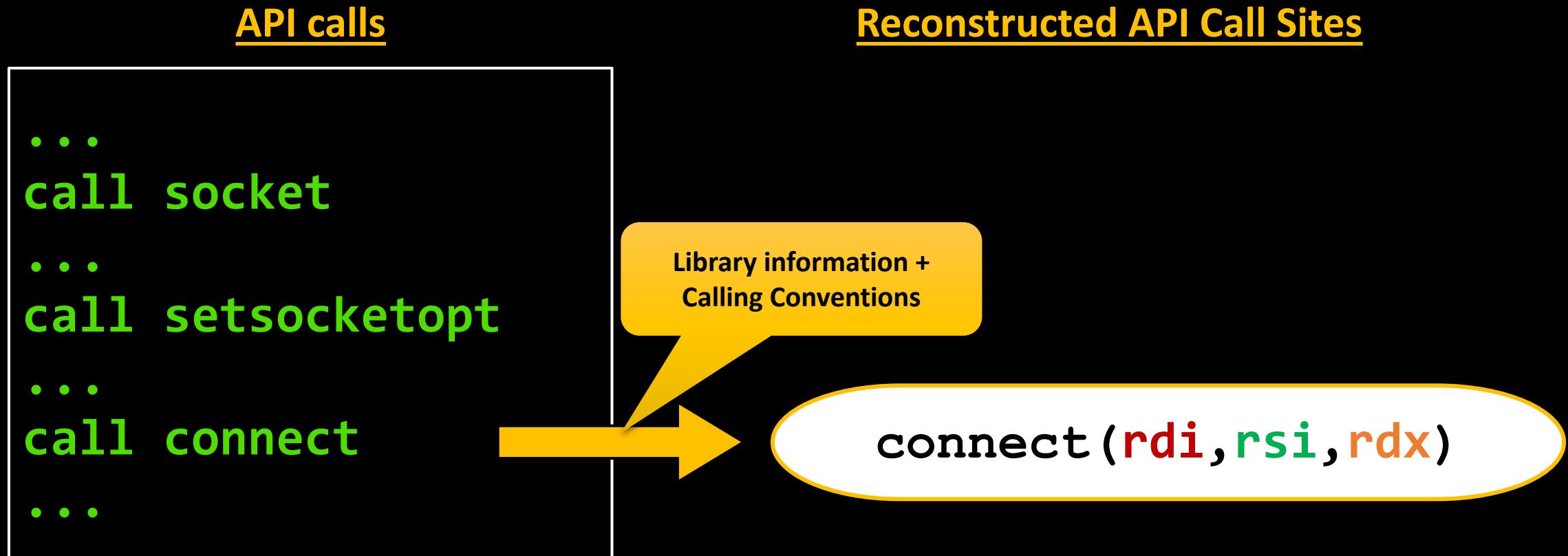


Our Approach

Extract Paths From the CFG



Using API Calls



Augmenting Call Sites

```
call  socket(...)  
mov   [rbp-58h],  rax  
mov   rax,  [rbp-58h]  
mov   rdi,  rax
```

```
mov   [rbp-50], rdi  
mov   rdi,  [rbp-50]  
mov   rsi,  rdi
```

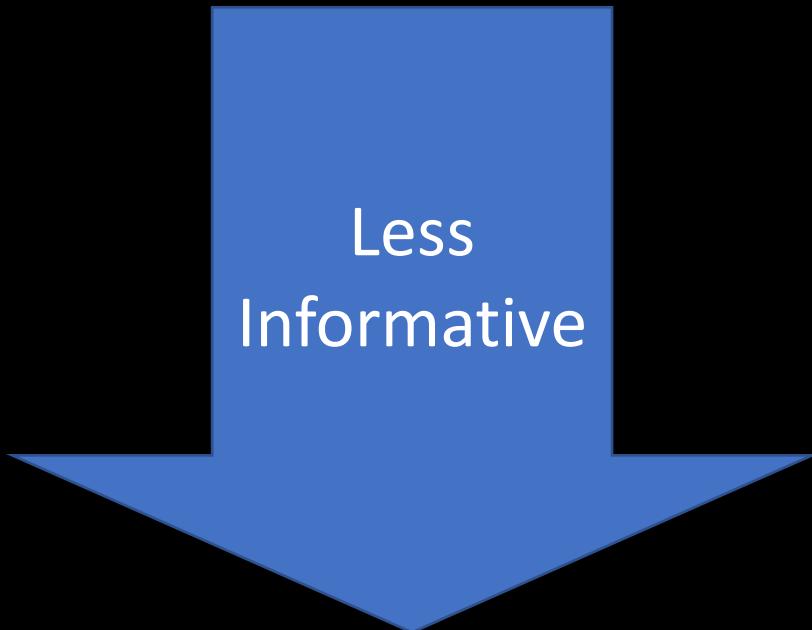
connect(**rdi**,**rsi**,**rdx**)

```
mov   rdx, 16
```

In the C code:
connect(sock,addr,16)

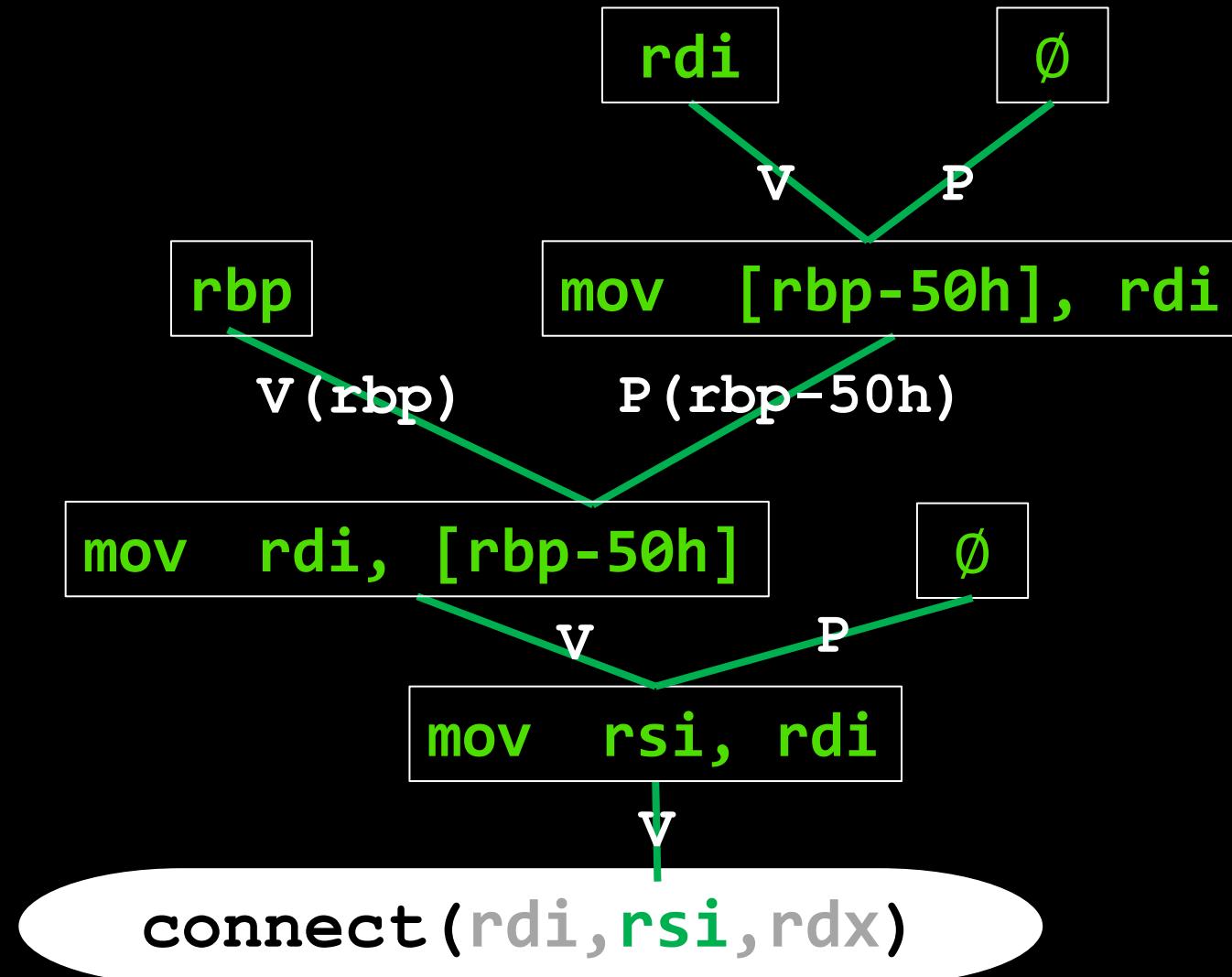
Using Concrete or Abstracted Values

1. Concrete value (`Integer`, `Enum`, `String`)
2. ARG – procedure argument
3. GLOBAL - pointer to a global variable
4. RET – a return value from a call
5. STACK – pointer to stack memory
6. \emptyset – no information

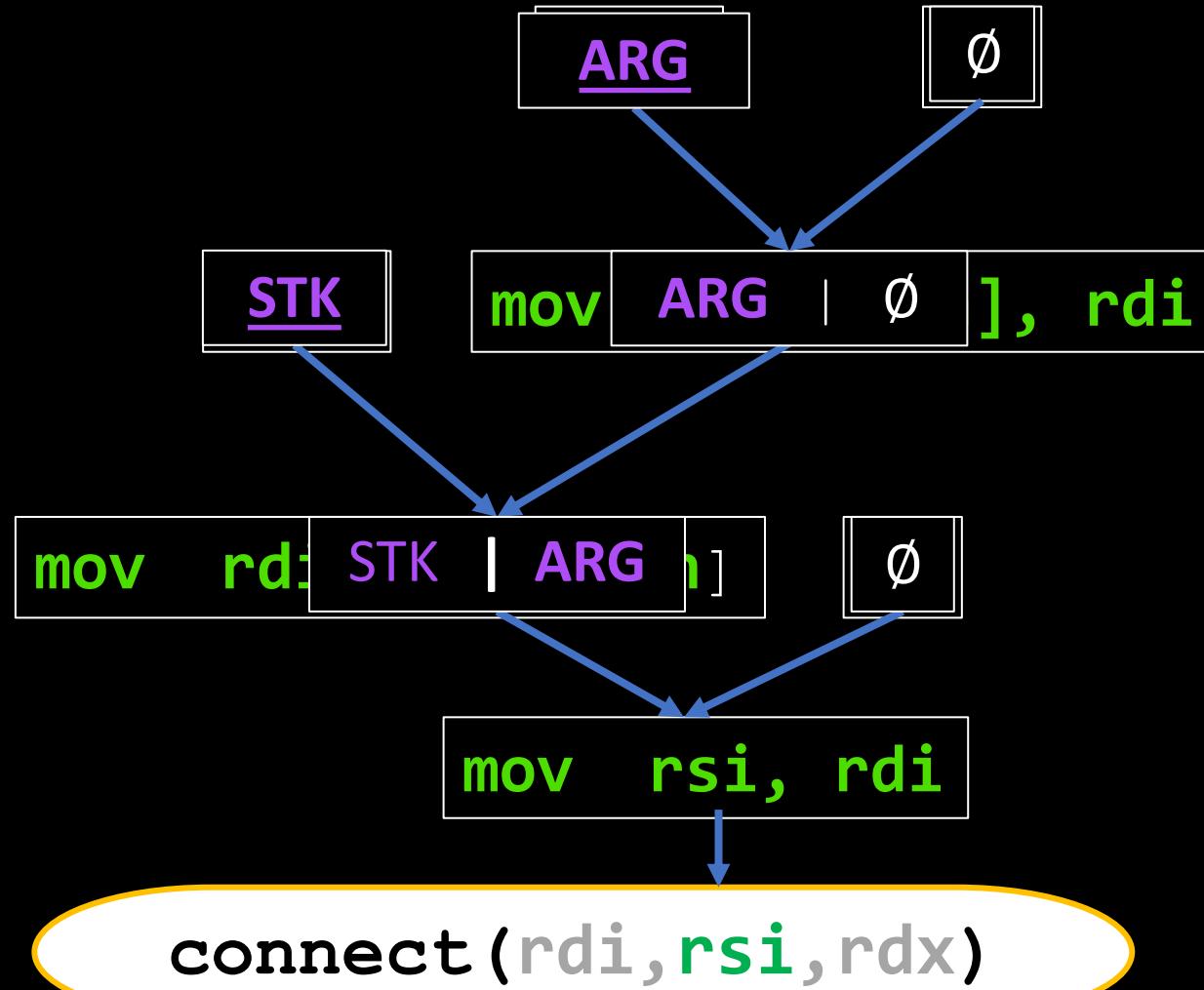


Less
Informative

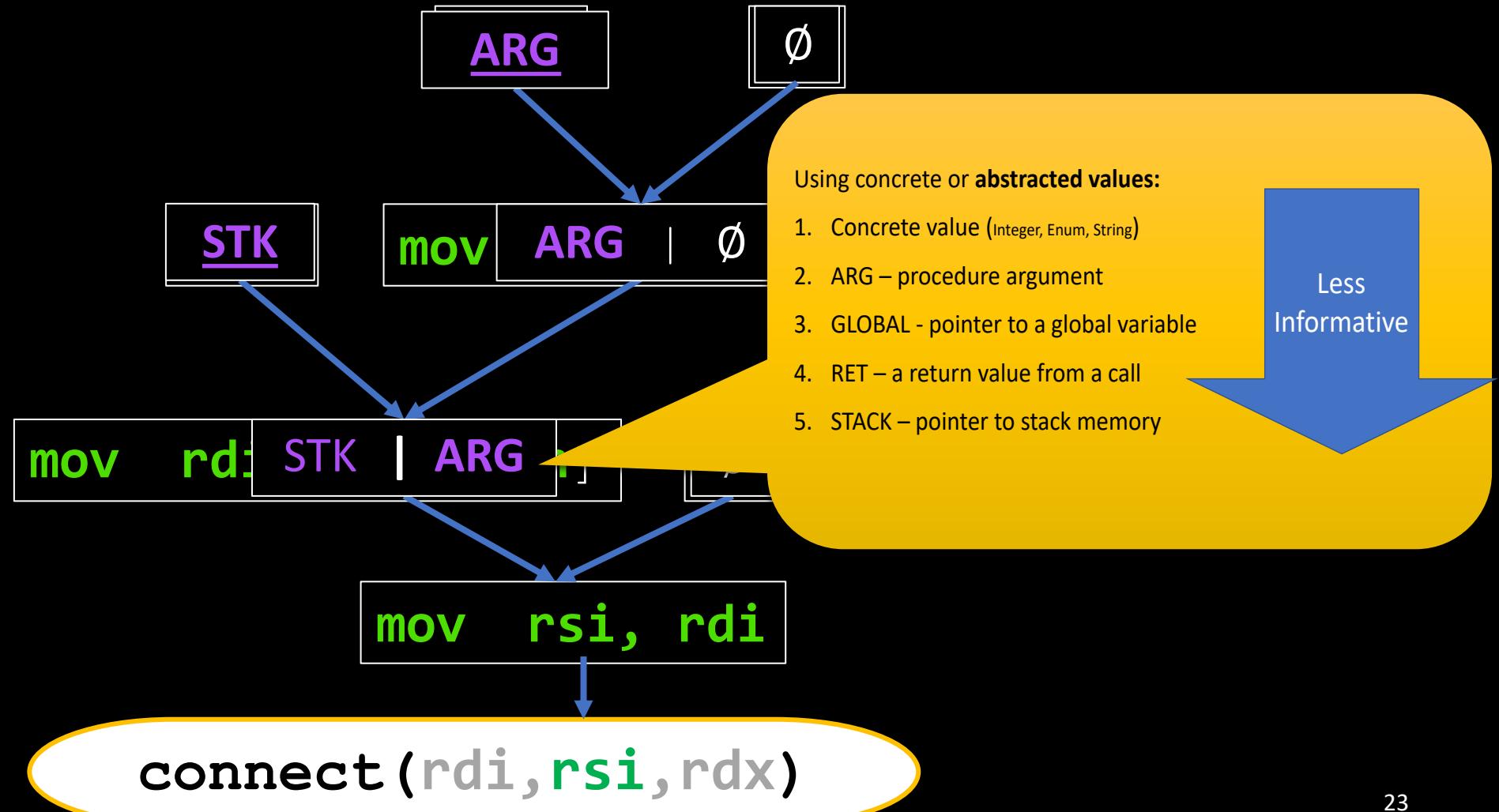
Pointer-Aware Slicing of Call Site Arguments



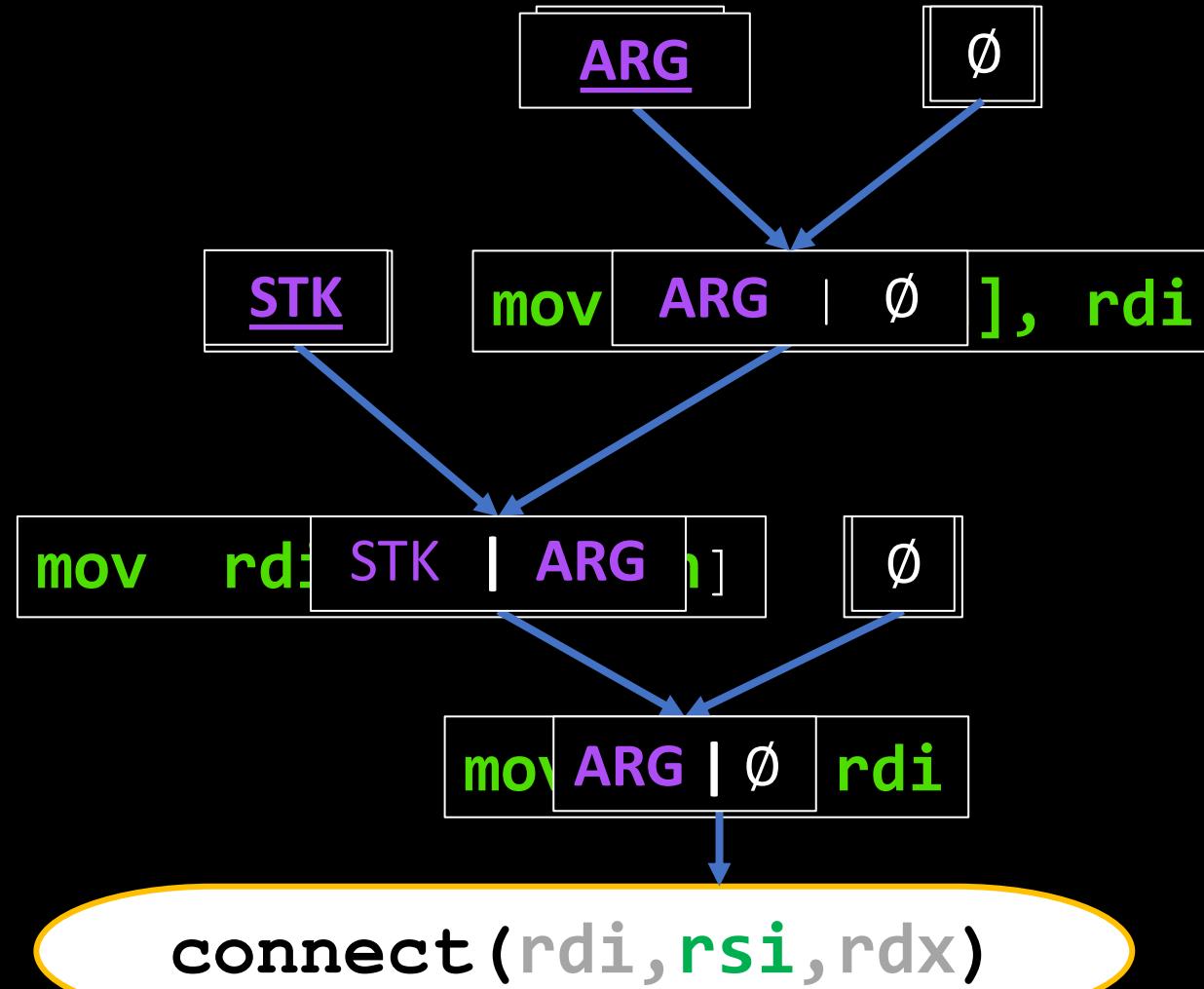
Augmenting Call Site Arguments



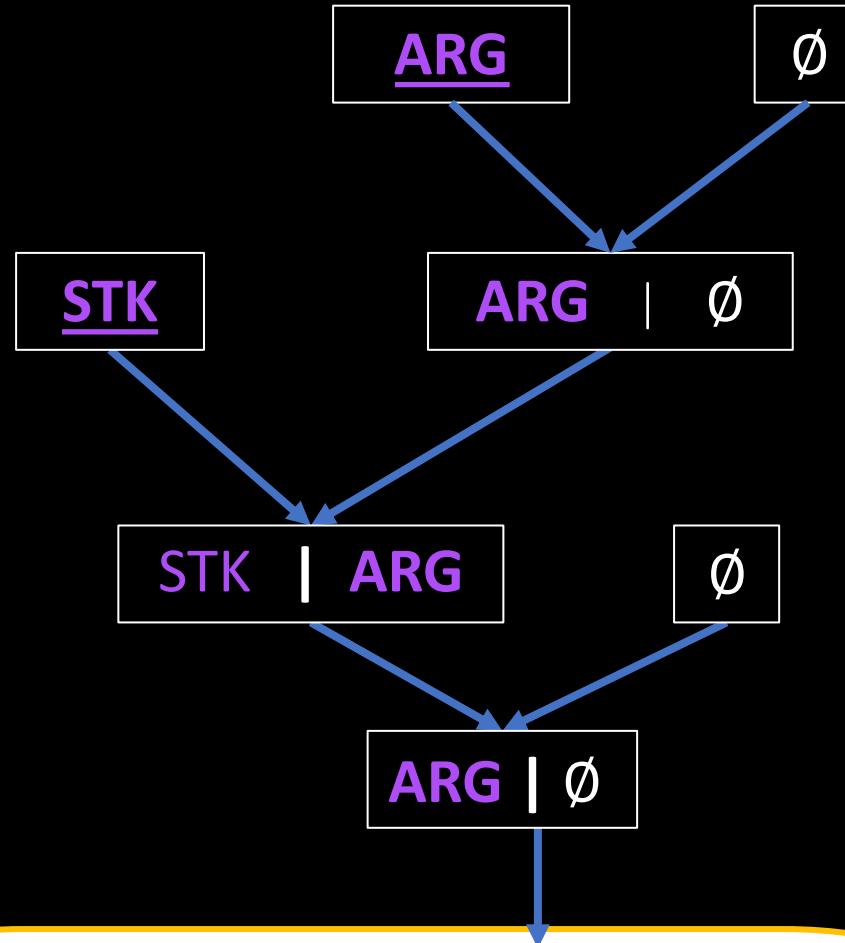
Augmenting Call Site Arguments



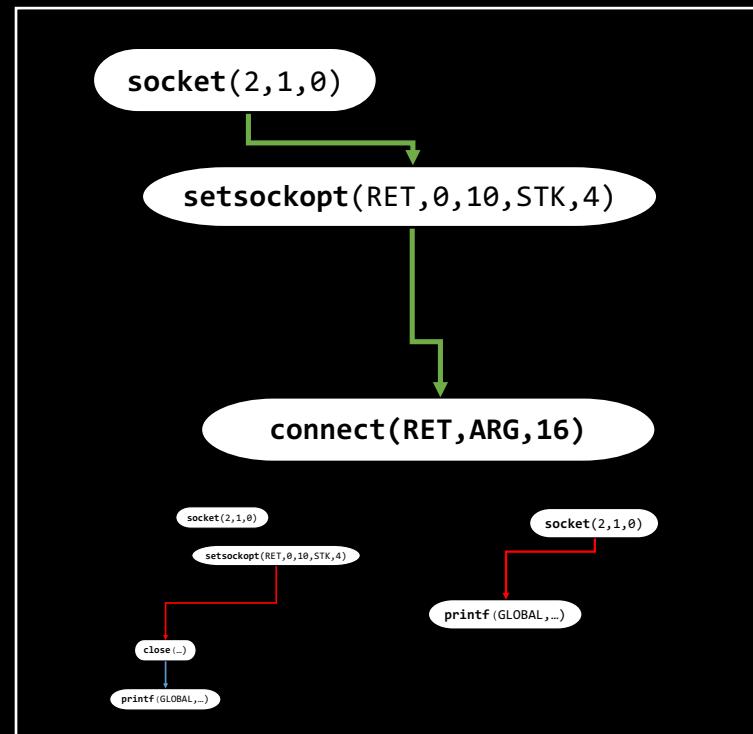
Augmenting Call Site Arguments



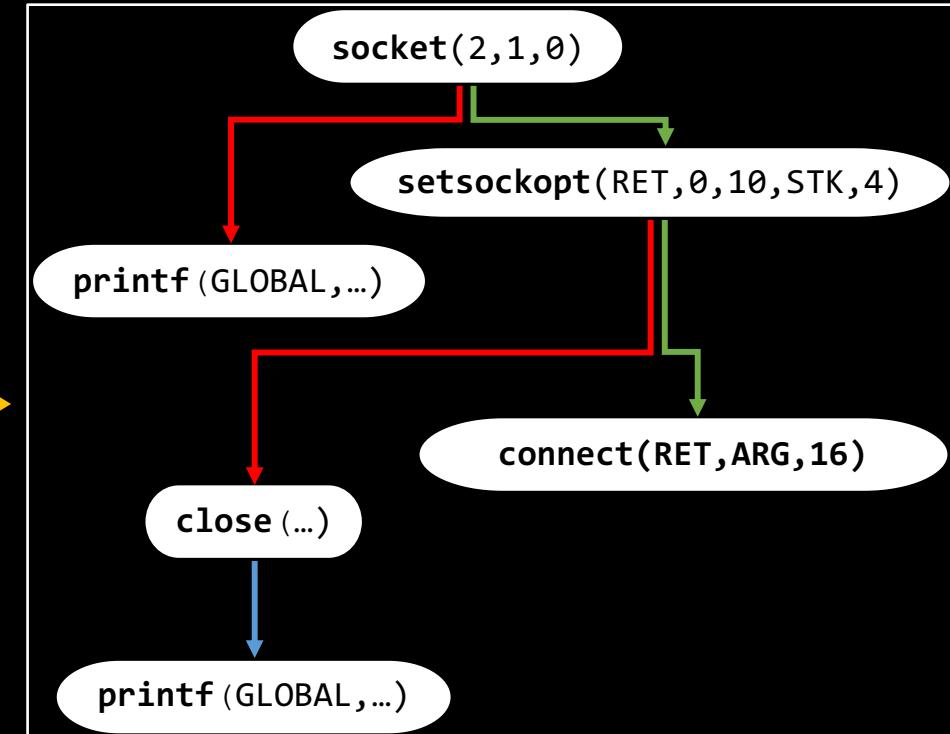
Augmenting Call Site Arguments



Augmented Control Flow Graph



Place augmented call sites
back in the CFG



LSTM & Transformer

GNN

Evaluation

Implementation: Nero



<https://github.com/tech-srl/Nero>

Evaluation Corpus

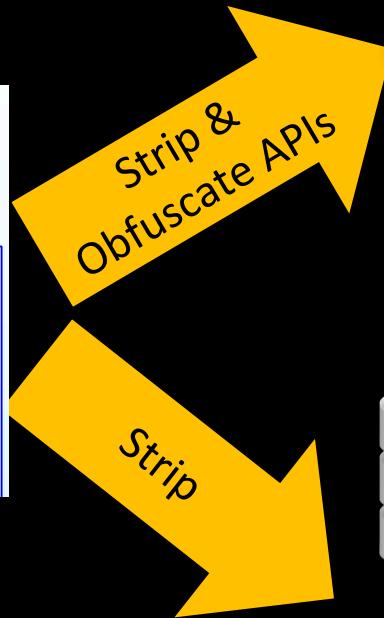


GNU software
repository

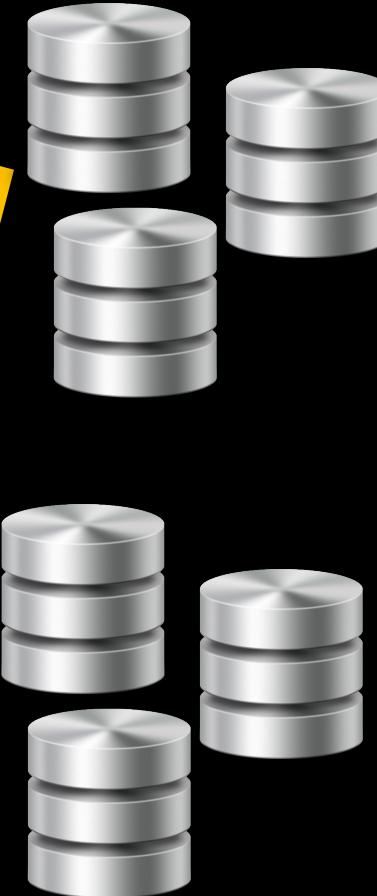
Cleanup

A screenshot of a debugger interface showing assembly code and call graphs. The assembly code includes instructions like `push rbp`, `mov rbp, rsp`, `sub rsp, 20h`, `mov [rbp-20], rdi`, and `short loc 436CDA`. Call graphs show the flow between procedures such as `sub_436B9D`, `loc_436C0A`, `loc_436B8B`, and `loc_436D93`.

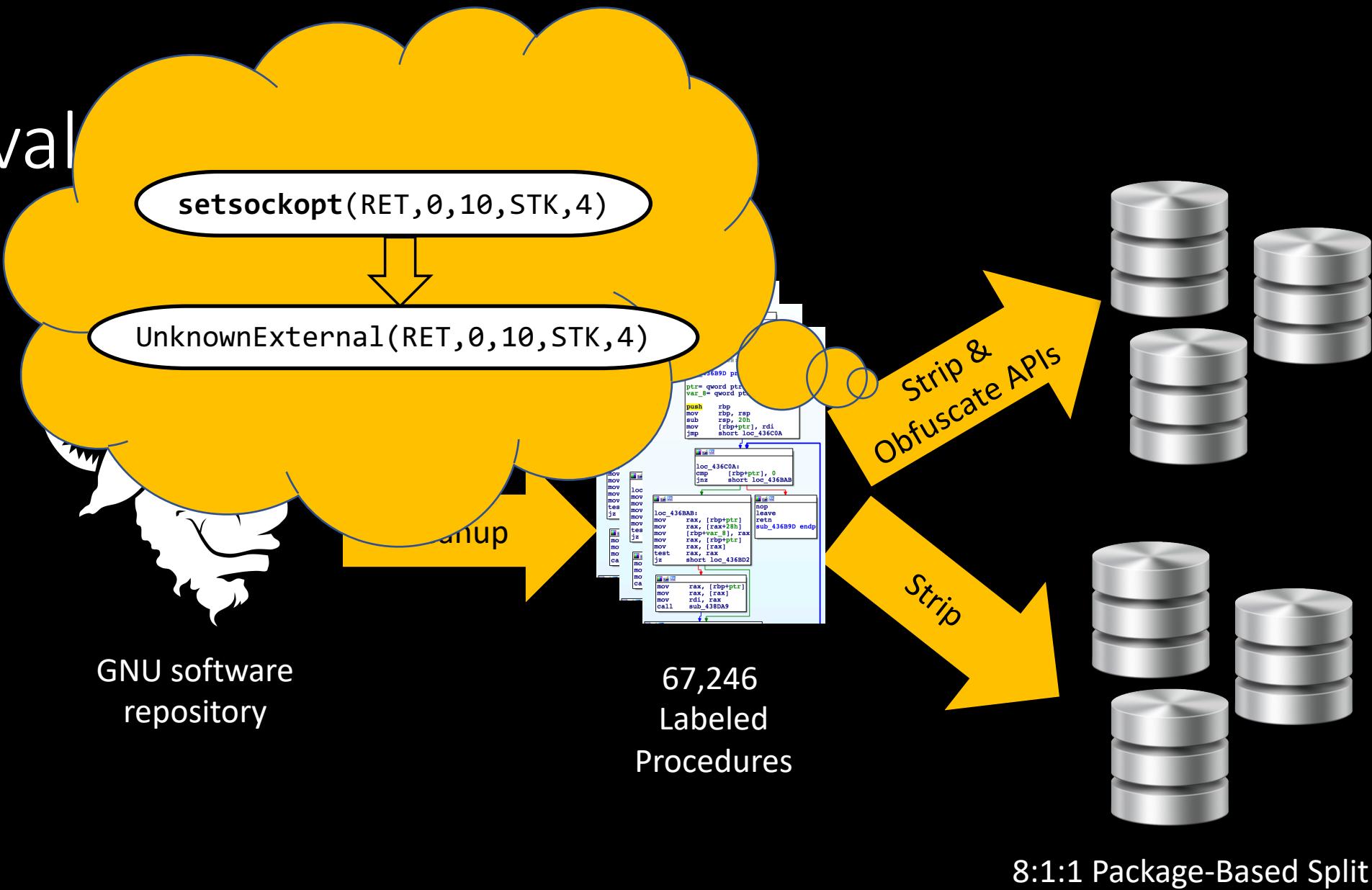
67,246
Labeled
Procedures



8:1:1 Package-Based Split



Evalu



Evaluation Results

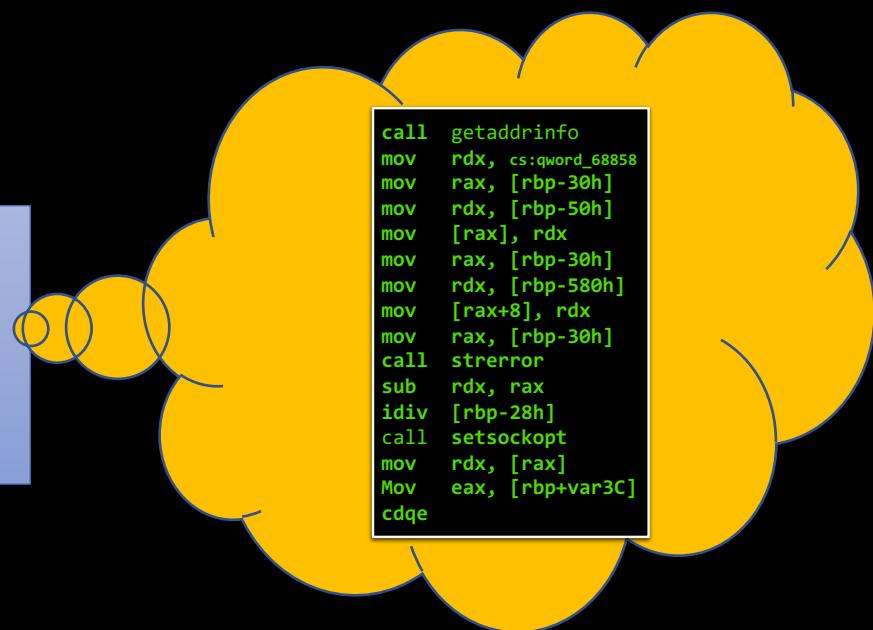
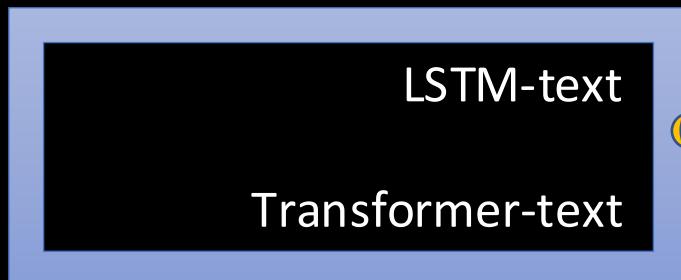
Nero-GNN

Nero-Transformer

Nero-LSTM

DIRE [Lacomis et al. 2019]

Debin [He et al. 2018]



Evaluation Results

Nero-GNN

Nero-Transformer

Nero-LSTM

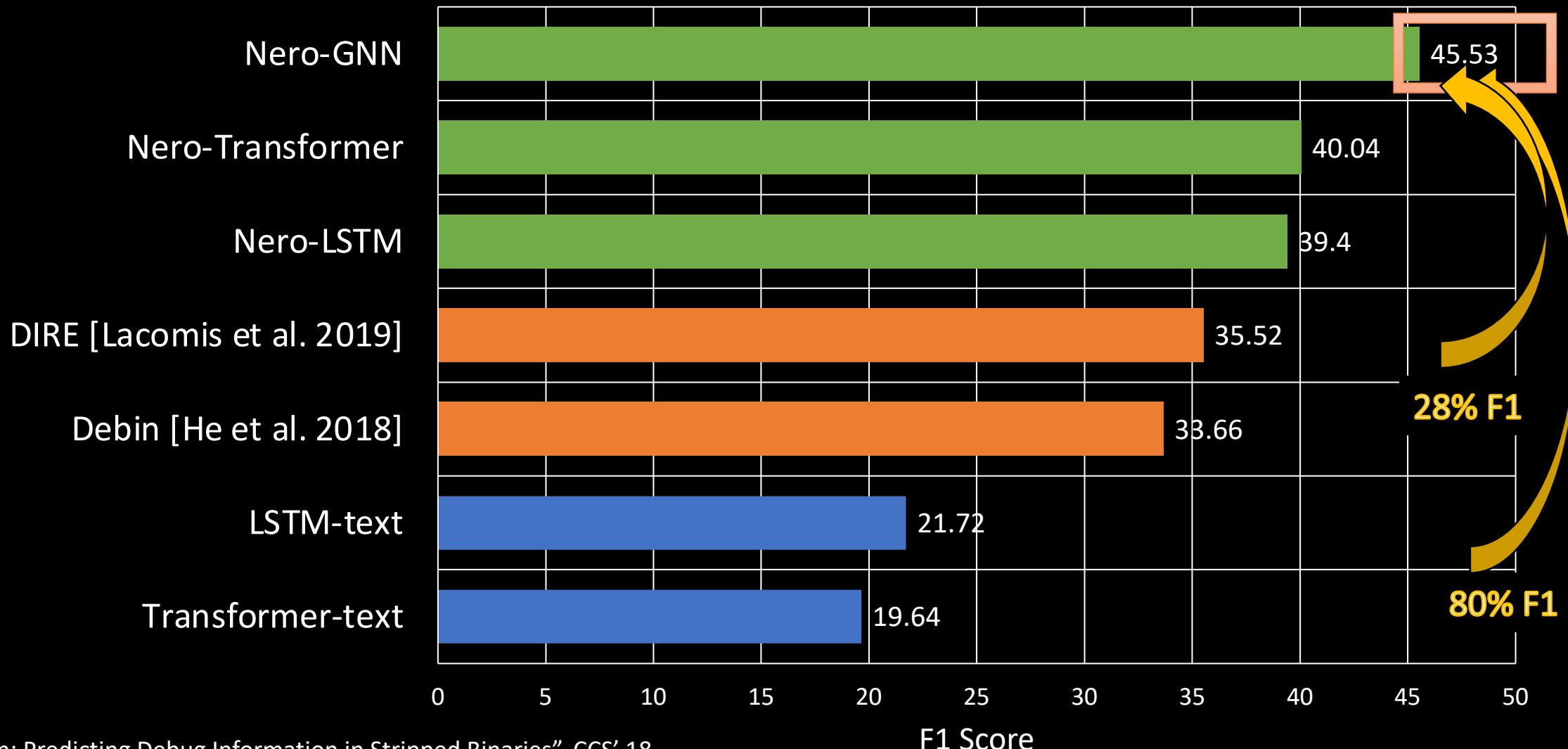
DIRE [Lacomis et al. 2019]

Debin [He et al. 2018]

LSTM-text

Transformer-text

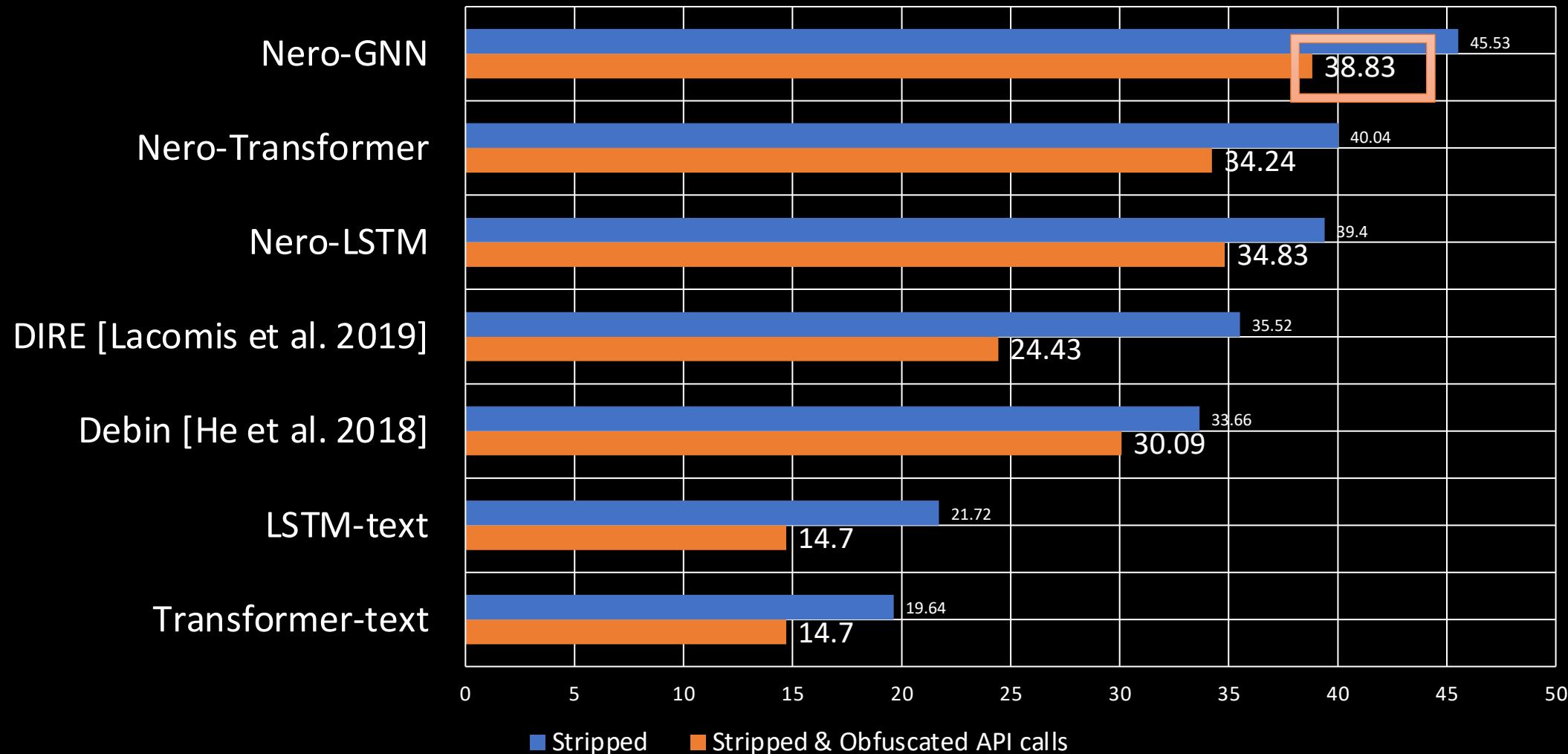
Evaluation Results



"Debin: Predicting Debug Information in Stripped Binaries", CCS' 18

"DIRE: A Neural Approach to Decompiled Identifier Naming", ASE '19

Evaluation Results

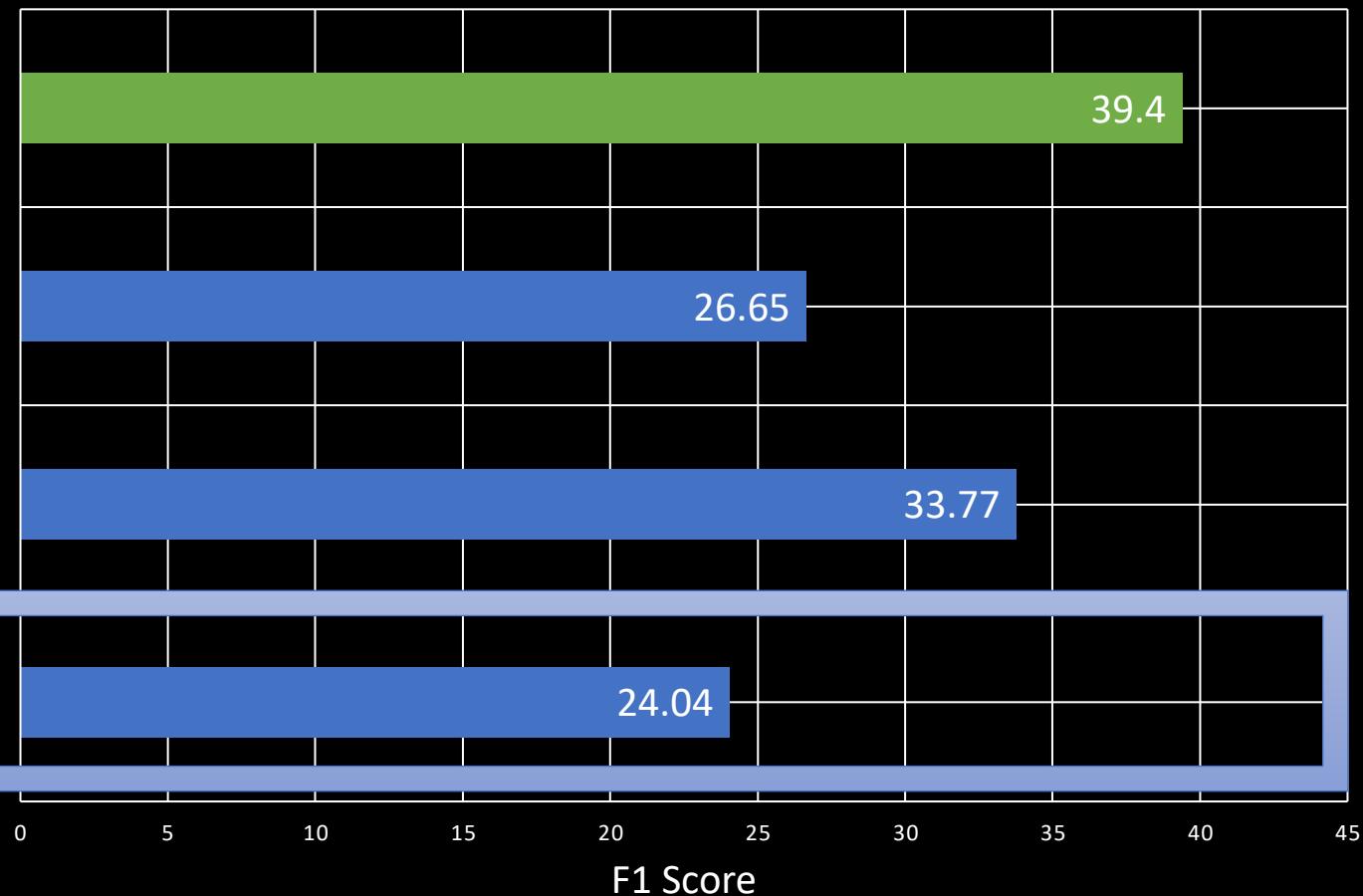


"Debin: Predicting Debug Information in Stripped Binaries", CCS' 18

"DIRE: A Neural Approach to Decompiled Identifier Naming", ASE '19

Ablation Study

Paths → Augmented Call Sites → LSTM



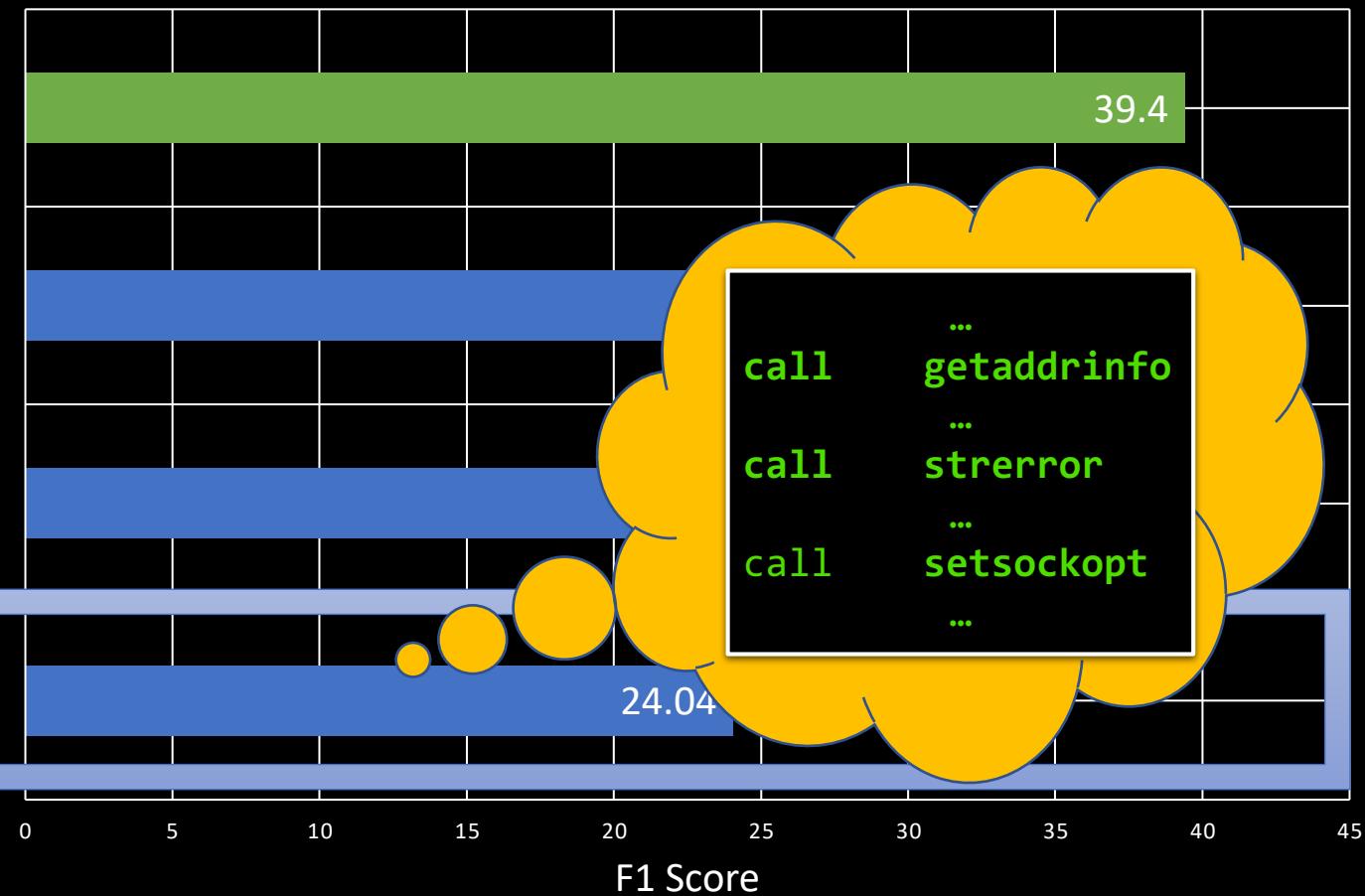
Paths → Only Calls → LSTM

Augmented Call Sites → LSTM

Only Calls → LSTM

Ablation Study

Paths → Augmented Call Sites → LSTM



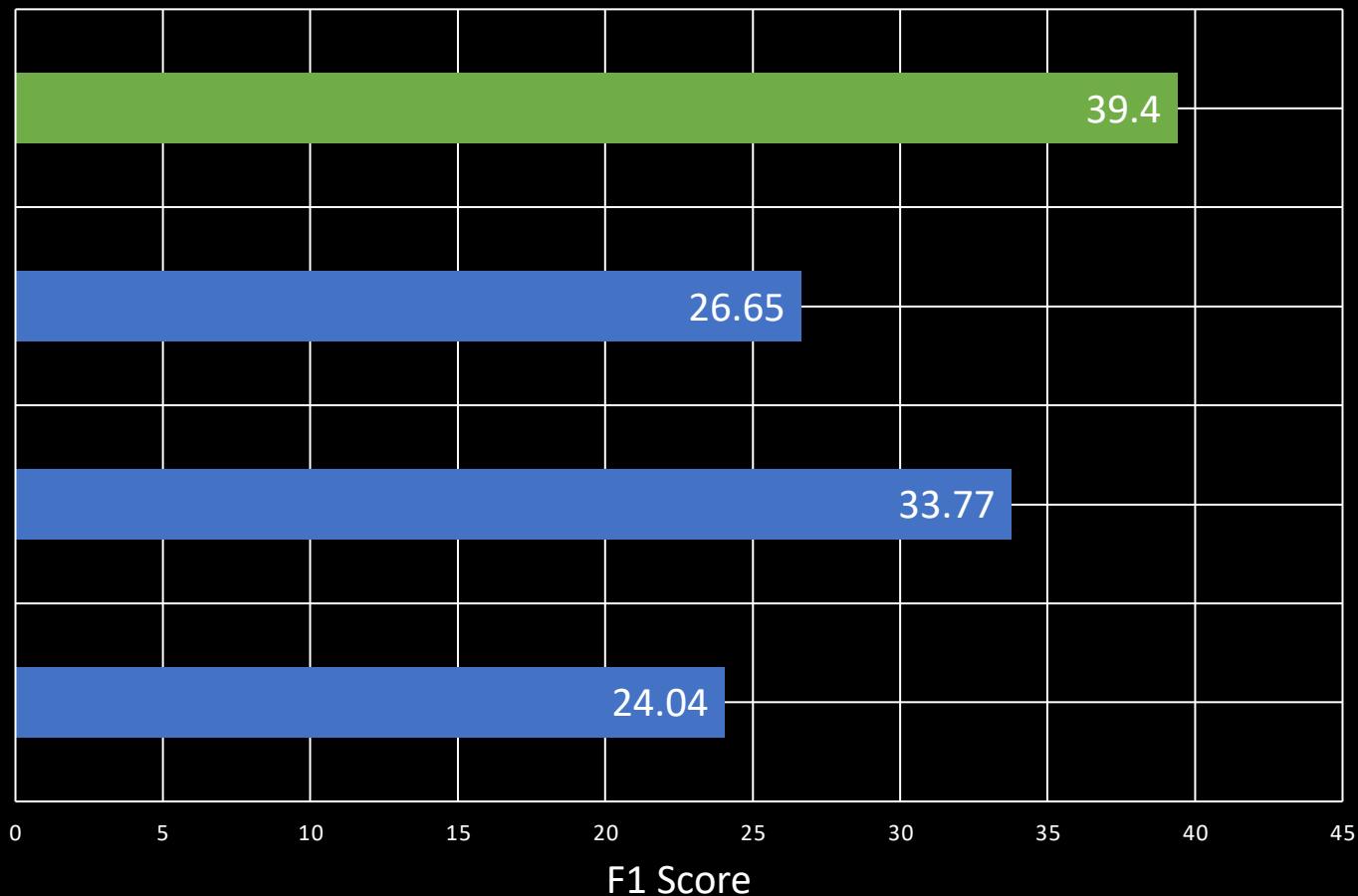
Paths → Only Calls → LSTM

Augmented Call Sites → LSTM

Only Calls → LSTM

Ablation Study

Paths → Augmented Call Sites → LSTM



Paths → Only Calls → LSTM

Augmented Call Sites → LSTM

Only Calls → LSTM

Qualitive Evaluation

Error Type	Ground Truth	Predicted Name
Programmers Vs English Language	i18n_initialize	i18n_init
	split_cfg_path	split_config_path
	add_env_opt	add_option
Date Structure Name Missing	get_best_speed	get_list_item
	ftp_parse_winnt_ls	parse_tree
	abort_gzip_signal	fatal_signal_handler
Verb Replaced	read_units	parse
	retrieve_from_file	get_from_file
	display_help	show_help

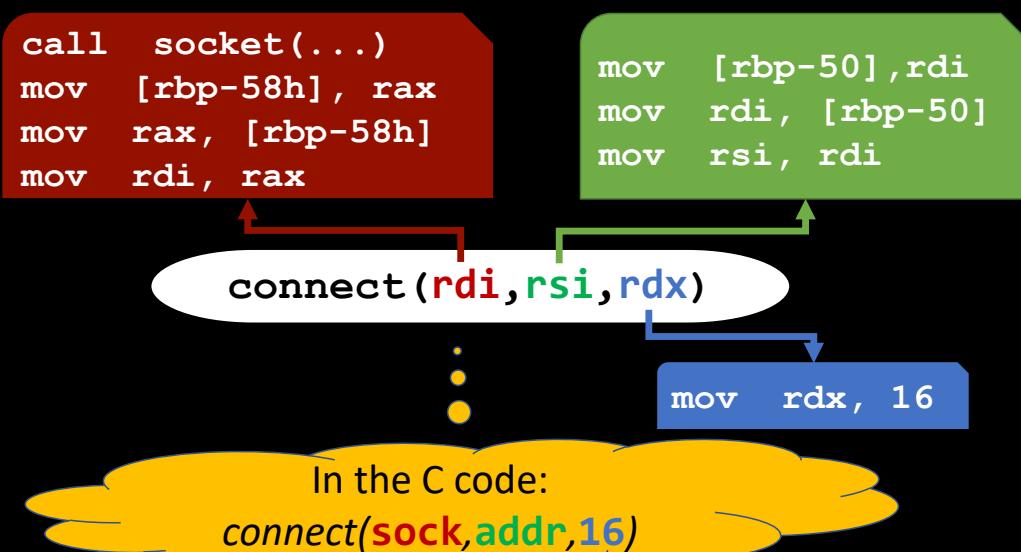
Qualitive Evaluation

Error Type	Ground Truth	Predicted Name
Programmers Vs English Language	i18n_initialize split_cfg_path add_env_opt	i18n_init split_item
Date Structure Name Missing	get_best_speed ftp_parse_winnt_ls abort_gzip_signal	parse_tree fatal_signal_handler
Verb Replaced	read_units retrieve_from_file display_help	parse get_from_file show_help

**Measured F1 is
actually a lower-
bound**

Takeaway Messages

Augmented call sites serve as a strong basis for procedure representations



Reconstructing the CFG enables the use of seq2seq and GNN models

