

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

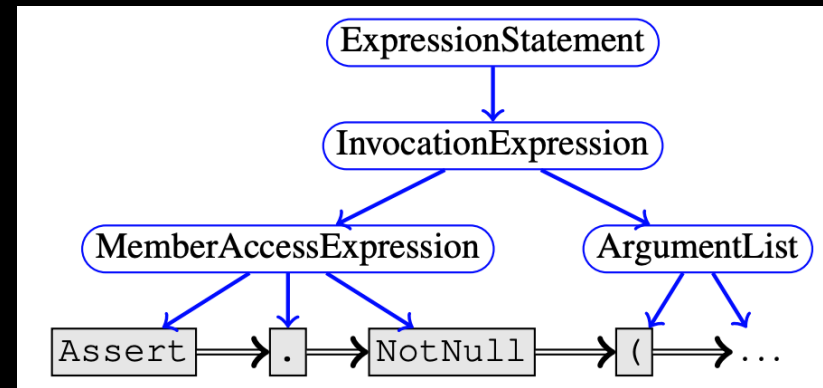
```

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

```

contains ^① ignore ^② case ^③

[“code2seq”, Alon et al., ICLR’2019]



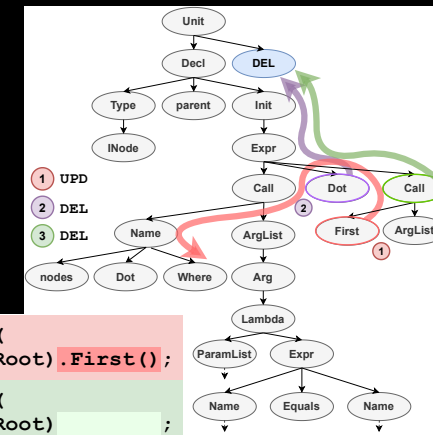
[Program graphs, Allamanis et al., ICLR’2018]

```

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

```

[“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

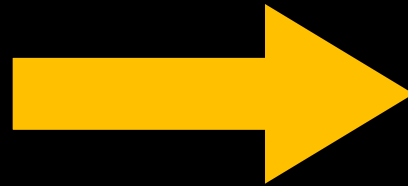
These put a lot of pressure on the model

- Optimization uses: entangled computational flows, context dependent

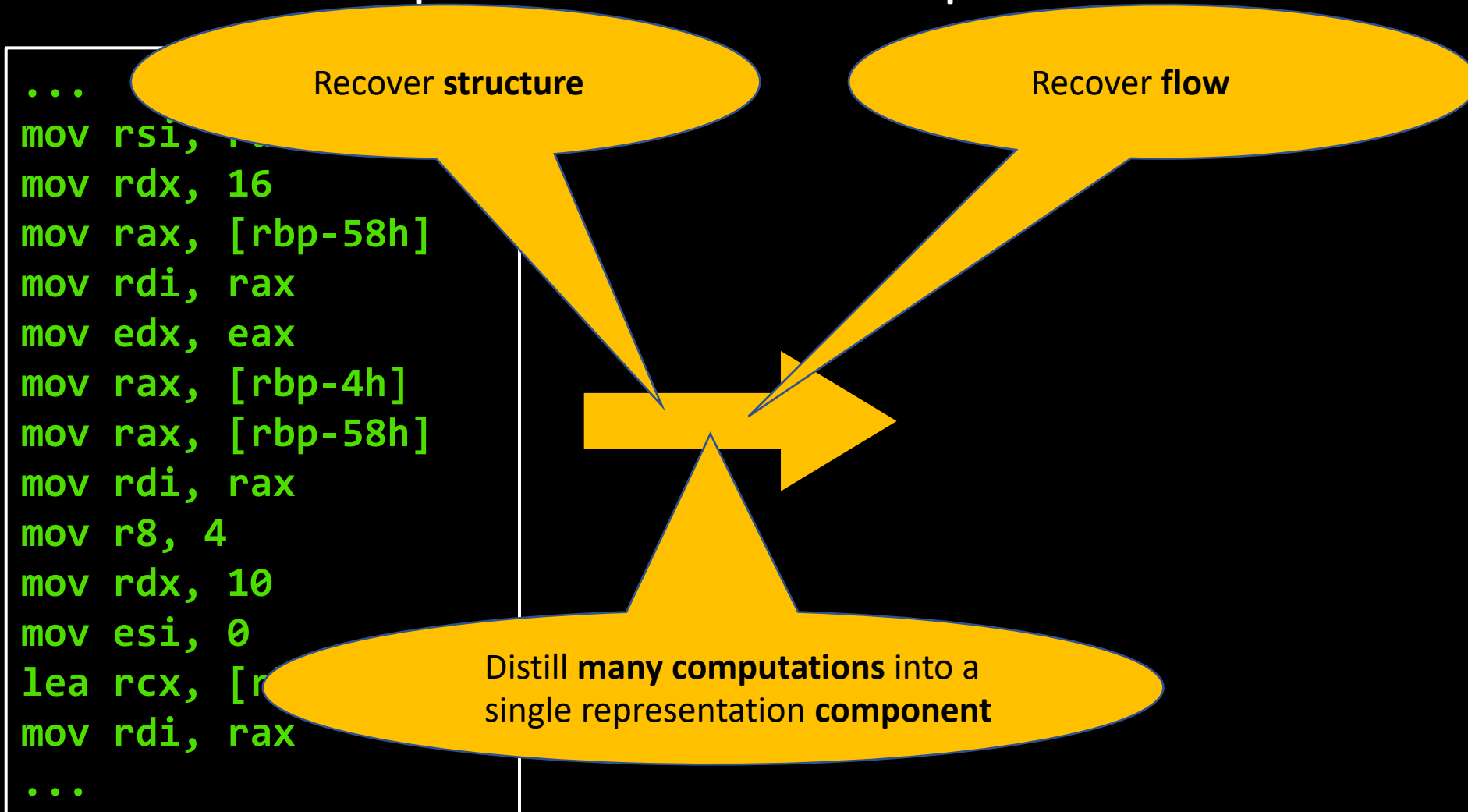
Long unstructured sequence of low-level operations

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  
...
```

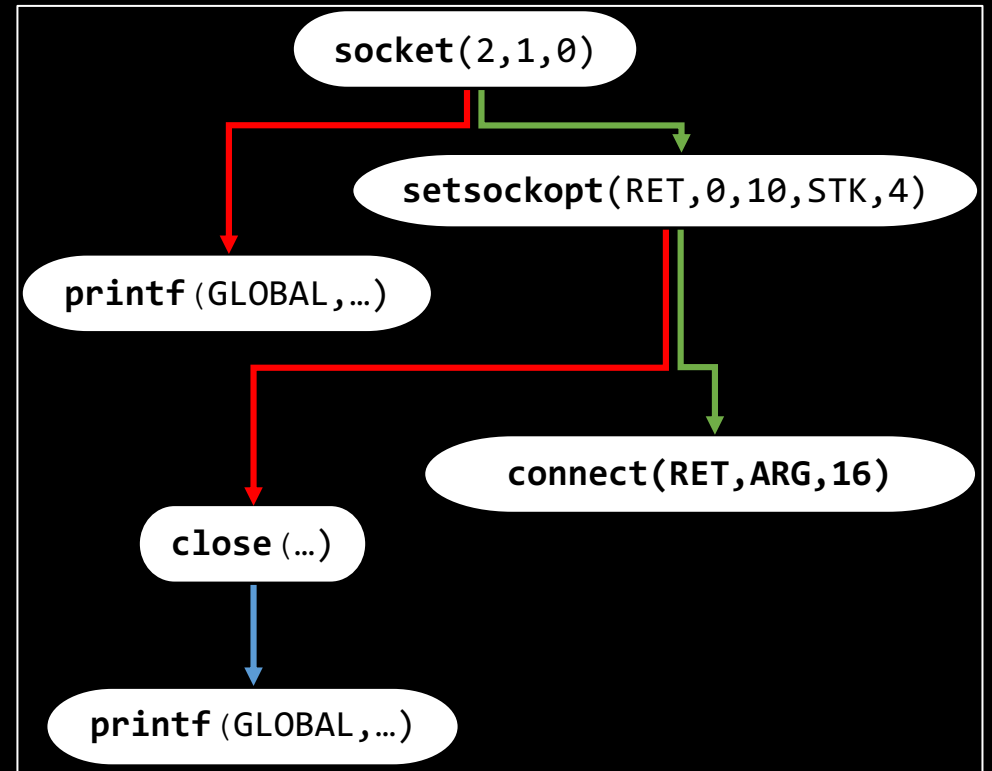
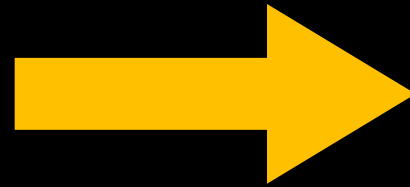


Key Idea: Use Binary Program Analysis to Create a Compact & Rich Representation

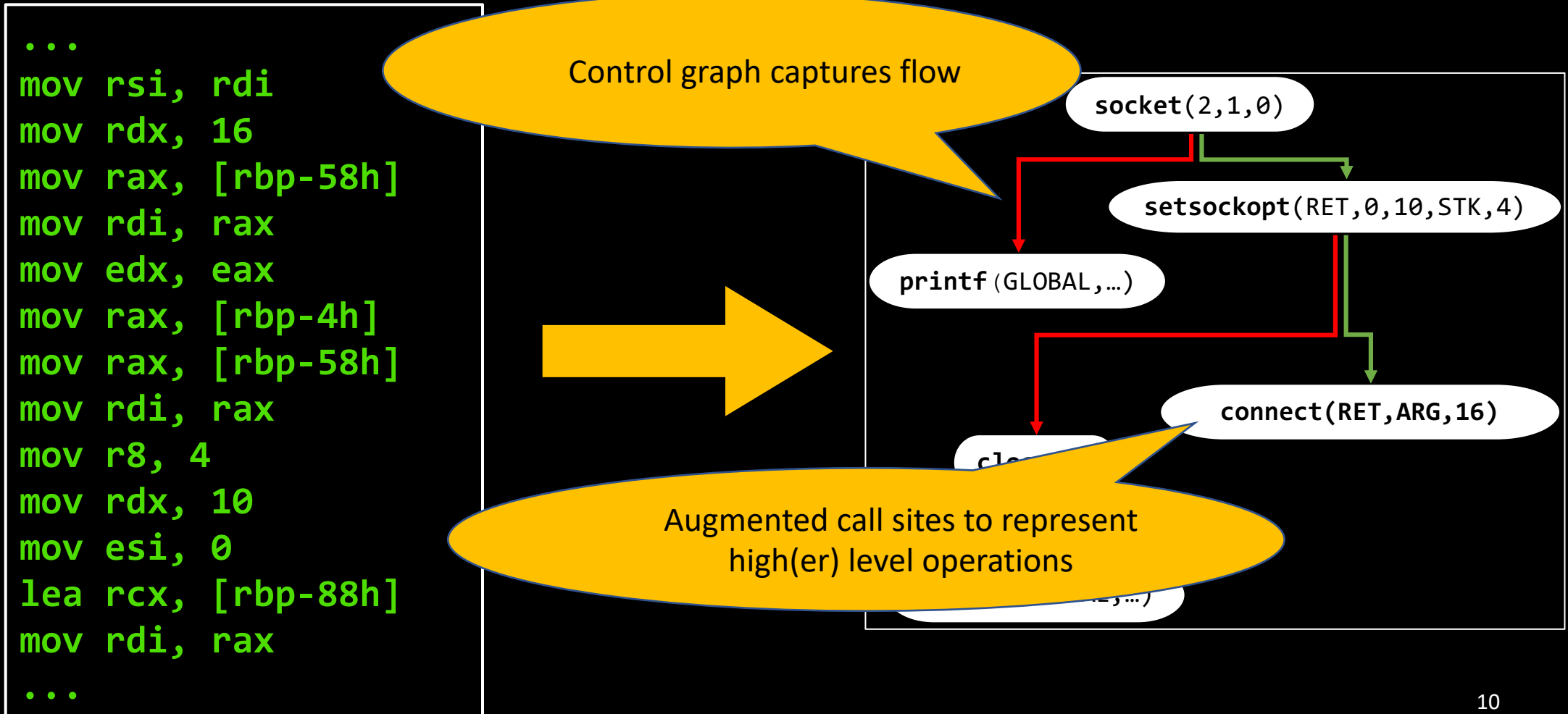


Key Idea: Use Binary Program Analysis to Create a Compact & Rich Representation

```
...  
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

| Function name |
|------------------------------|
| f sub_42EFA7 |
| f sub_42F08D |
| f sub_42F0C0 |
| f sub_42F5EA |
| f sub_42F680 |
| f sub_42F717 |
| f sub_42F7F8 |
| f sub_42F809 |
| f sub_42F863 |
| f sub_42F878 |
| f sub_42F8CC |
| f sub_42F901 |
| f sub_42FAC8 |
| f sub_42FDE3 |
| f sub_42FE25 |
| f sub_42FF77 |
| f sub_430094 |
| f sub_4305BE |
| f sub_4307C1 |
| f sub_430873 |
| f sub_430A13 |
| f sub_430A88 |

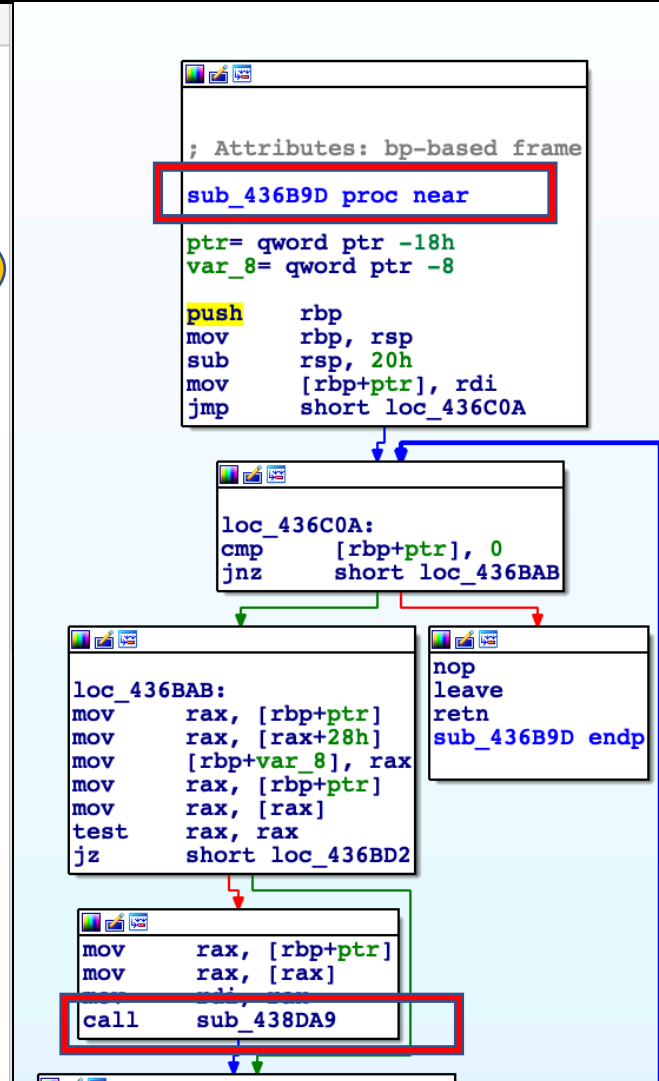
```
graph TD
    sub_436B9D["sub_436B9D proc near  
ptr= qword ptr -18h  
var_8= qword ptr -8  
push rbp  
mov rbp, rsp  
sub rsp, 20h  
mov [rbp+ptr], rdi  
jmp short loc_436C0A"]
    loc_436C0A["loc_436C0A:  
cmp [rbp+ptr], 0  
jnz short loc_436BAB"]
    loc_436BAB["loc_436BAB:  
mov rax, [rbp+ptr]  
mov rax, [rax+28h]  
mov [rbp+var_8], rax  
mov rax, [rbp+ptr]  
mov rax, [rax]  
test rax, rax  
jz short loc_436BD2"]
    nop["nop  
leave  
retn  
sub_436B9D endp"]
    call_block["mov rax, [rbp+ptr]  
mov rax, [rax]  
call sub_438DA9"]

    sub_436B9D --> loc_436C0A
    loc_436C0A --> loc_436BAB
    loc_436C0A --> nop
    loc_436BAB --> call_block
```

Helping Reverse Engineers

| Function name |
|--|
|  sub_42EFA7 |
|  sub_42F08D |
|  sub_42E0C0 |
|  sub_42F800 |
|  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 |

Where to start?



Un-Stripping Procedure Names

| Function name | Function name |
|------------------------------|--|
| f sub_42EFA7 | f limit_bandwidth |
| f sub_42F08D | f write_data |
| f sub_42F0C0 | f fd_read_body |
| f sub_42F5EA | f fd_read_hunk |
| f sub_42F680 | f line_terminator |
| f sub_42F717 | f fd_read_line |
| f sub_42F7F8 | f retr_rate |
| f sub_42F809 | f calc_rate |
| f sub_42F863 | f retrieve_url |
| f sub_42F878 | f retrieve_from_file |
| f sub_42F8CC | f printwhat |
| f sub_42F901 | f sleep_between_retrievals |
| f sub_42FAC8 | f free_urlpos |
| f sub_42FDE3 | f rotate_backups |
| f sub_42FE25 | f getproxy |
| f sub_42FF77 | f url_uses_proxy |
| f sub_430094 | f no_proxy_match |
| f sub_4305BE | f set_local_file |
| f sub_4307C1 | f input_file_url |
| f sub_430873 | f spider_cleanup |
| f sub_430A13 | f nonexistent_url |
| f sub_430A88 | f print_broken_links |
| | f url_unescape_1 |
| | f url_unescape |
| | f url_unescape_except_reserved |
| | f url_escape_1 |

```
graph TD
    Entry["; Attributes: bp-based frame  
public free_urlpos  
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"]
    Entry --> Loc436C0A["loc_436C0A:  
cmp [rbp+ptr], 0  
jnz short loc_436BAB"]
    Loc436C0A --> Loc436BAB["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"]
    Loc436C0A --> EndBlock["nop  
leave  
retn  
free_urlpos endp"]
    Loc436BAB --> Call["mov rax, [rbp+ptr]  
mov rax, [rax]  
mov rdi, rax  
call url_free"]
    Call --> EndBlock
```

Un-Stripping Procedure Names

The image shows a debugger interface with a list of functions on the left and assembly code on the right. A yellow callout bubble points to the function name 'free_urlpos' in the list, with the text 'Start at the right place'. Another yellow callout bubble points to a 'call url_free' instruction in the assembly code, with the text 'Focus on the right procedure'. The assembly code for 'free_urlpos' includes instructions for stack frame setup, pushing registers, and a call to 'url_free'.

Function name

- sub_42F5E...
- sub_42F680
- sub_42F717
- sub_42F7F8
- sub_42F809
- 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

Function name

- free_urlpos
- rotate_backups
- getproxy
- spider_clean...
- nonexisting_url
- print_broken_links
- url_unescape_1
- url_unescape
- url_unescape_except_reserved
- url_escape_1

```
; Attributes: bp-based frame
public free_urlpos
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

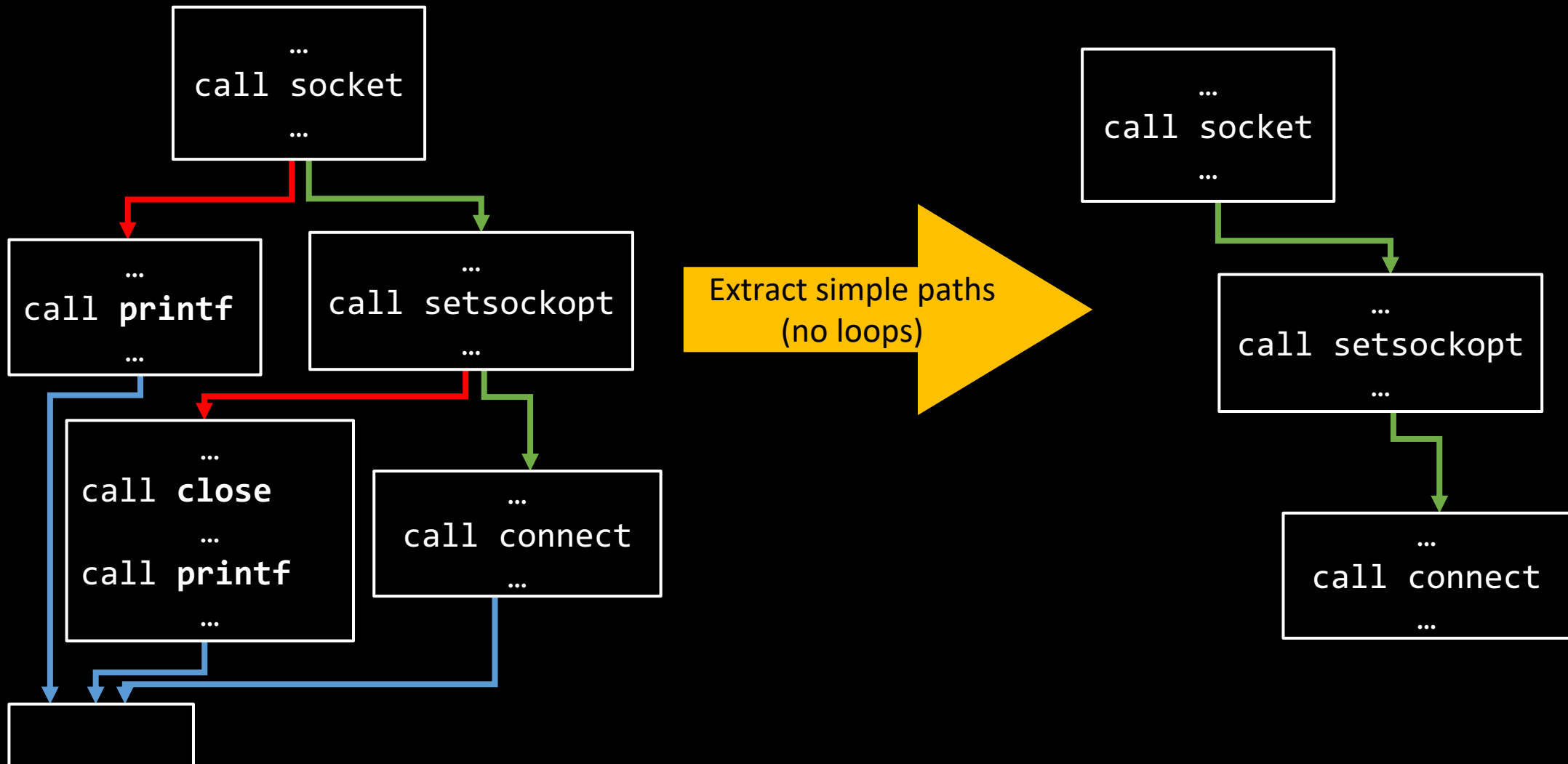
loc_436C0A:
cmp [rbp+ptr], 0
jnz short loc_436BAB

nop
leave
retn
free_urlpos endp

mov rax, [rbp+ptr]
mov rax, [rax]
mov rdi, rax
call url_free
```

Our Approach

Extract Paths From the CFG



Using API Calls

API calls

```
...  
call socket  
...  
call setsockopt  
...  
call connect  
...
```

Library information +
Calling Conventions

Reconstructed API Call Sites

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

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)`

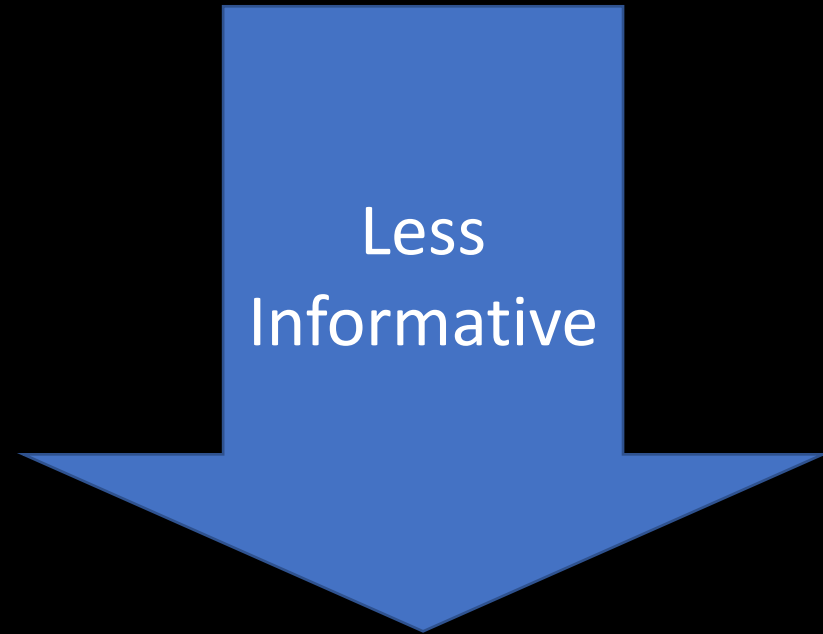
```
mov   rdx, 16
```

In the C code:

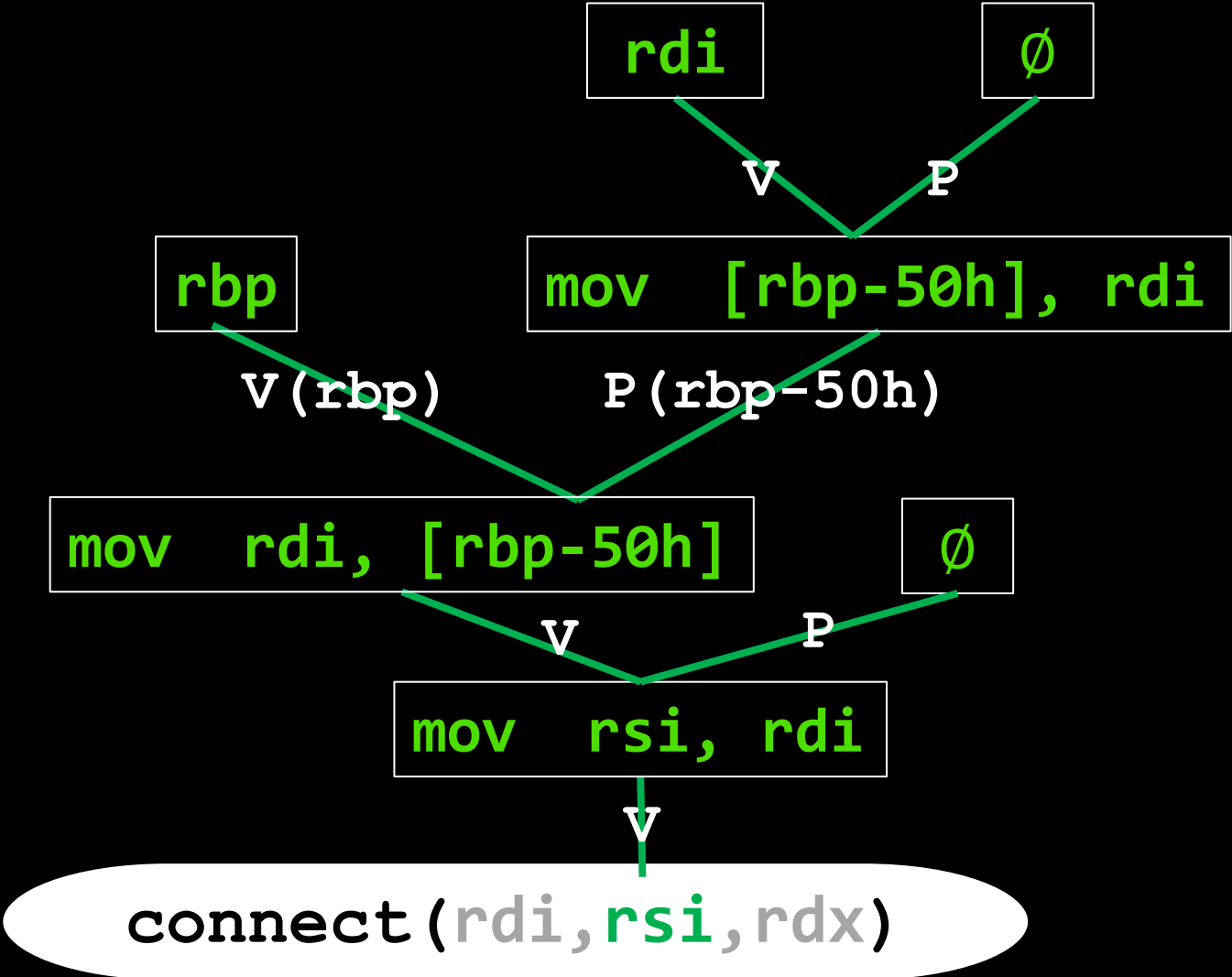
```
connect(sock, addr, 16)
```

Using Concrete or Abstracted Values

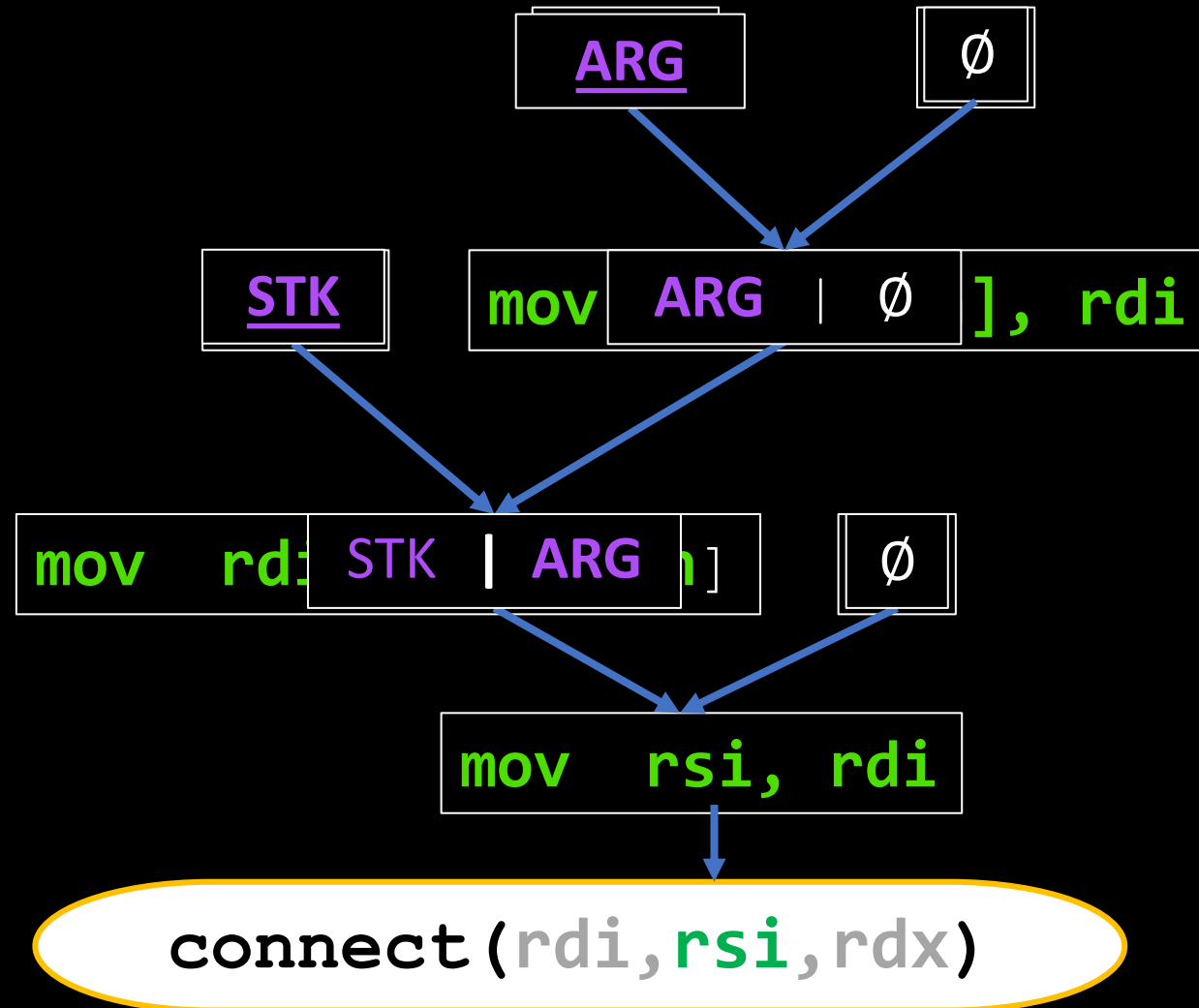
1. Concrete value (Integer, Enum, String)
2. ARG – procedure argument
3. GLOBAL - pointer to a global variable
4. RET – a return value from a call
5. STACK – pointer to stack memory
6. \emptyset – no information



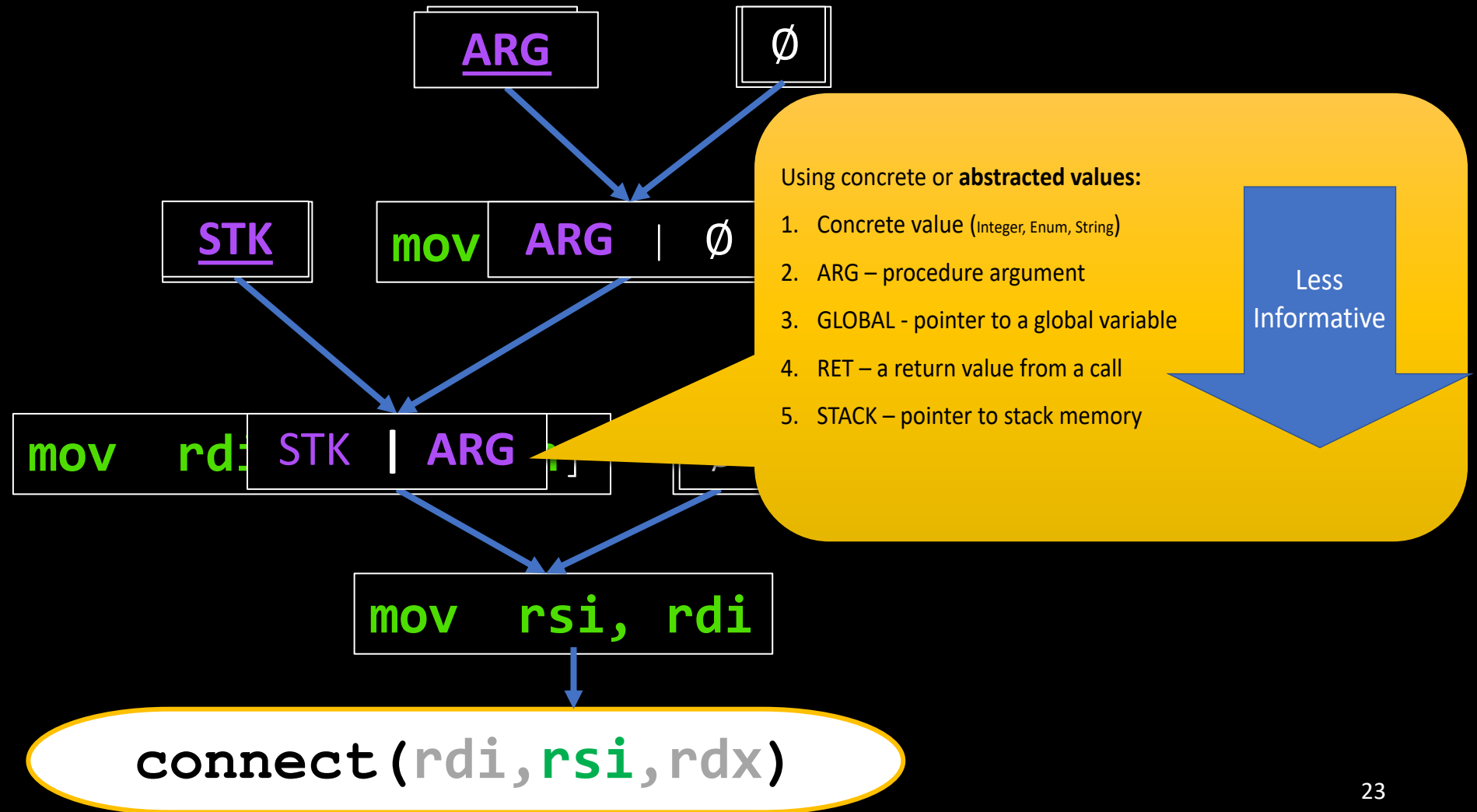
Pointer-Aware Slicing of Call Site Arguments



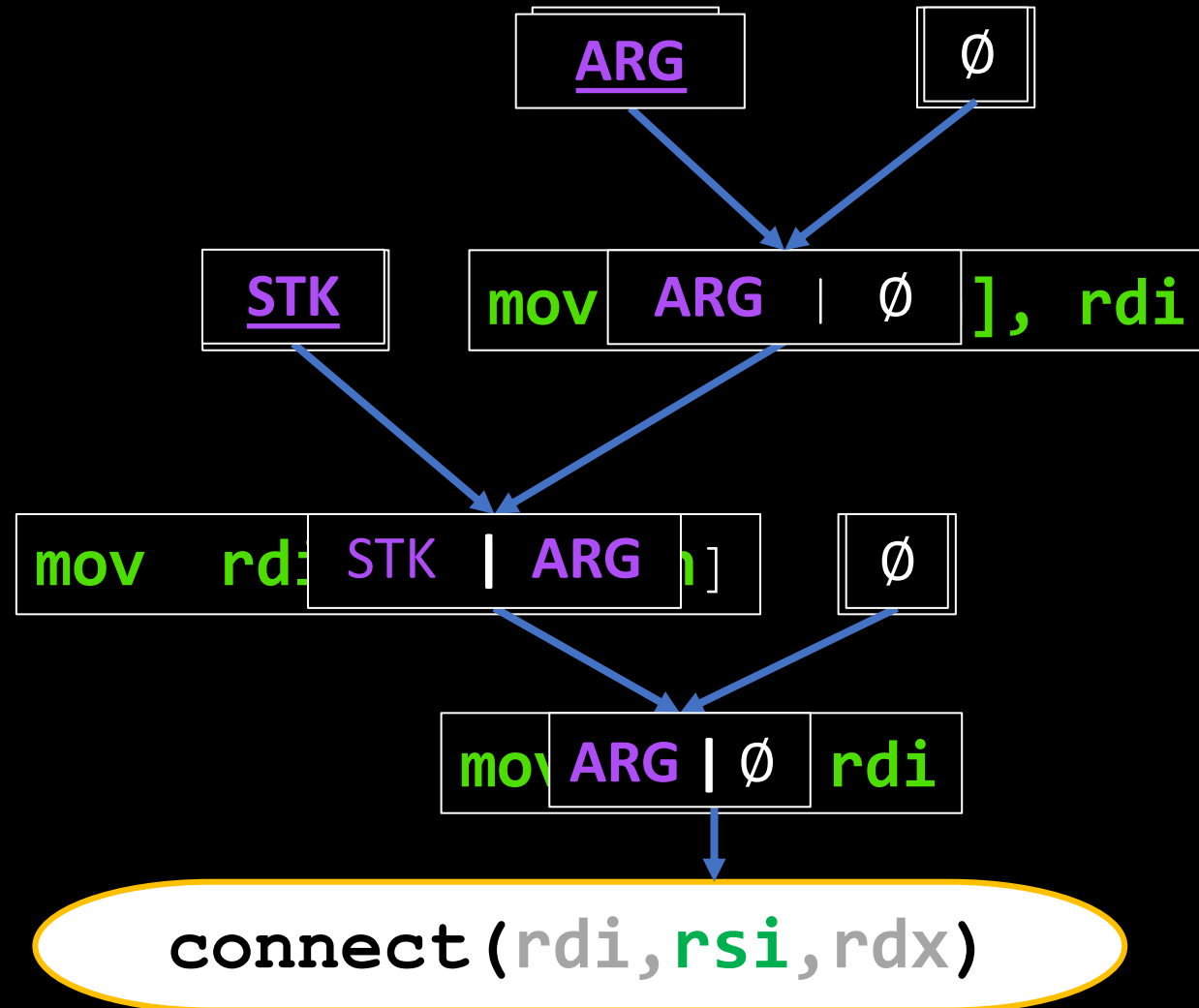
Augmenting Call Site Arguments



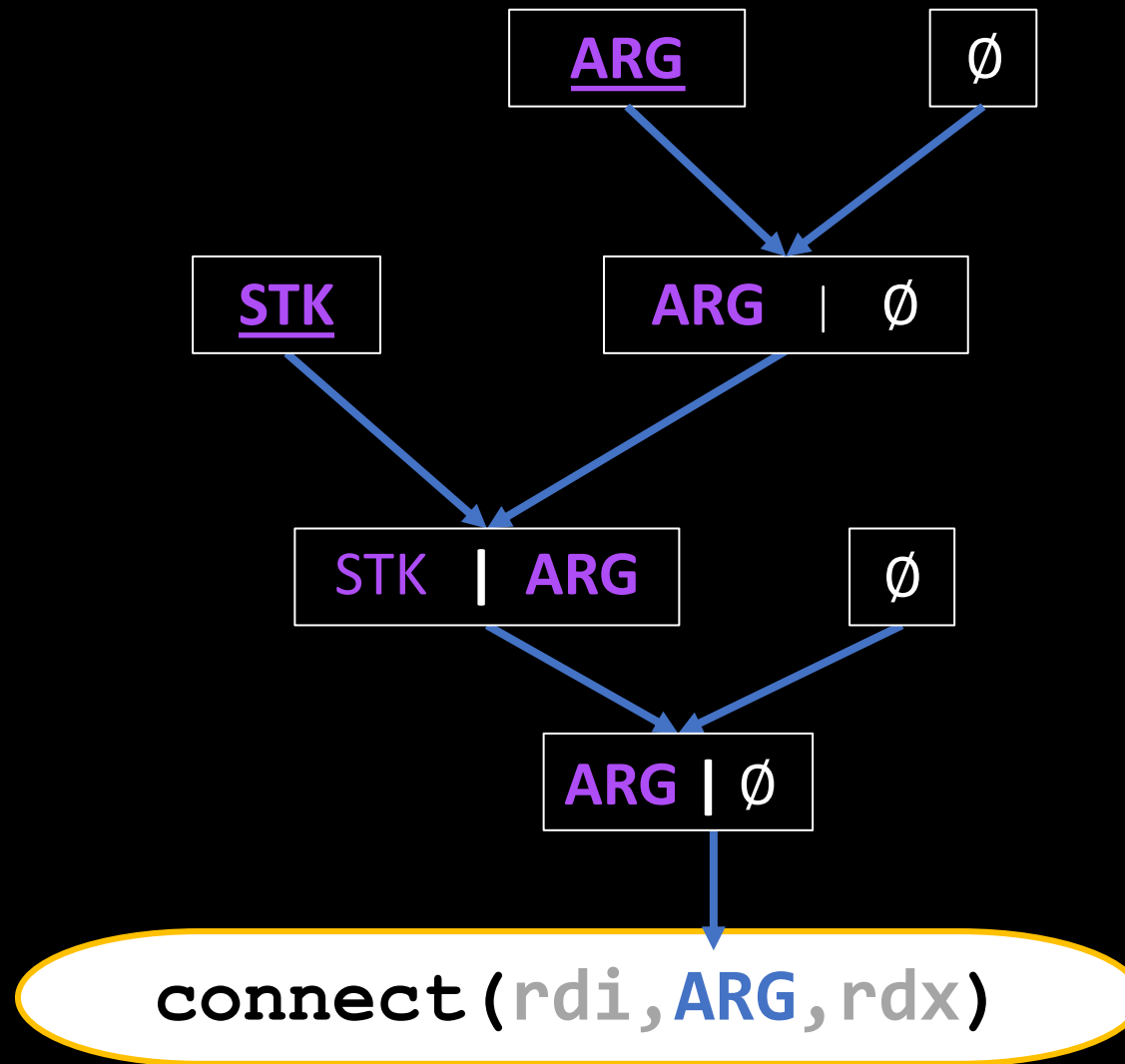
Augmenting Call Site Arguments



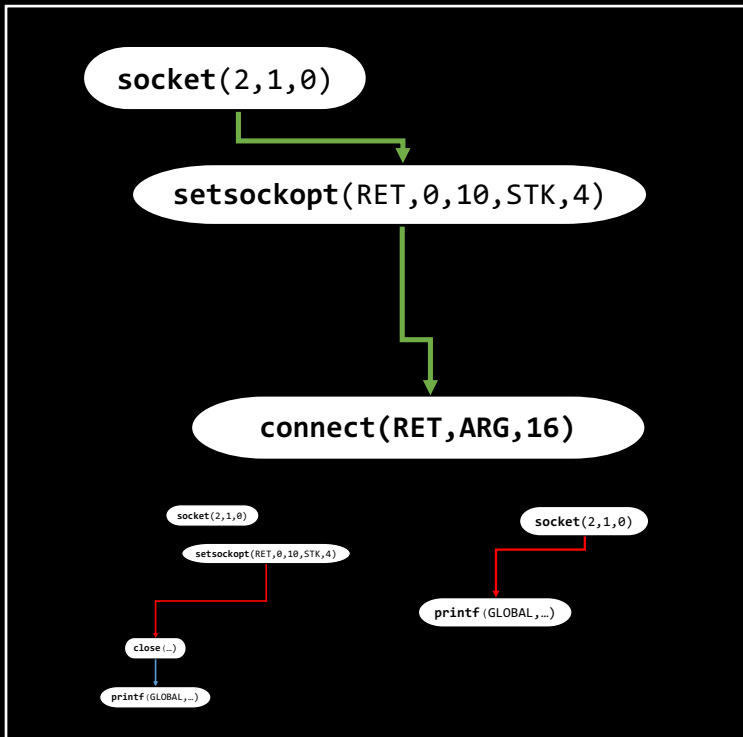
Augmenting Call Site Arguments



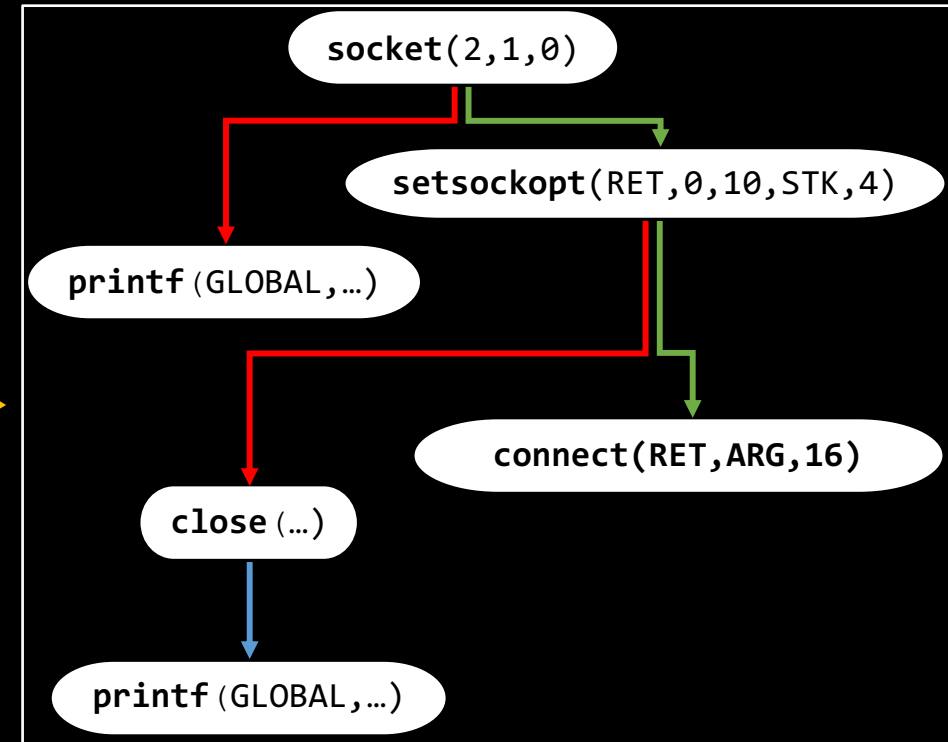
Augmenting Call Site Arguments



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

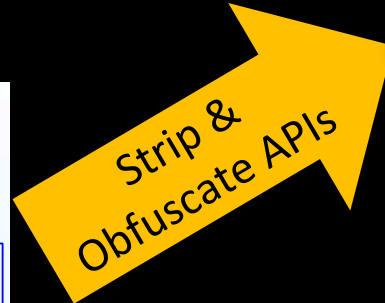
```
; Attributes: bp-based frame
sub_436B9D proc near
ptr= qword ptr -18h
var_8= qword ptr -8
push rbp
mov rbp, rsp
sub rsp, 20h
mov [rbp+ptr], rdi
jmp short loc_436C0A

loc_436C0A:
cmp [rbp+ptr], 0
jns short loc_436BAA

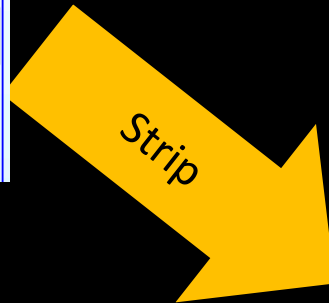
loc_436BAA:
nop
leave
mov rax, [rax+28h]
mov [rbp+var_8], rax
test rax, [rax]
mov rax, [rax]
test rax, rax
jz short loc_436BD2
ca

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

67,246
Labeled
Procedures



Strip &
Obfuscate APIs

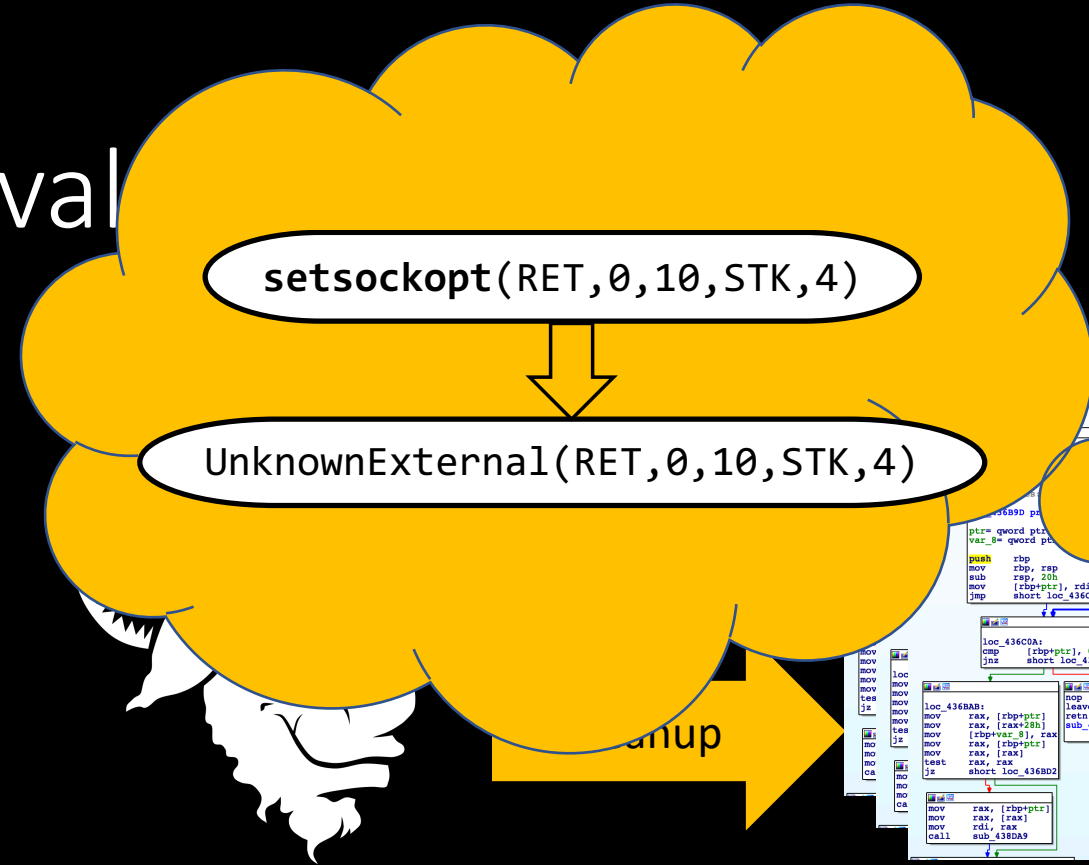


Strip



8:1:1 Package-Based Split

Eval



GNU software repository

```
pt= qword ptr [rbp+pt]
var_8= qword ptr [rbp+var_8]
push rbp
mov rbp, rsp
sub rsp, 20h
mov [rbp+ptr], rdi
jmp short loc_436C0A

loc_436C0A:
cmp [rbp+ptr], 0
jns short loc_436BAA

loc_436BAA:
nop
leave
ret

loc_436B9D:
mov rax, [rbp+ptr]
mov rax, [rax+28h]
sub_436B9D endp

loc_436B8D:
mov rax, [rbp+var_8]
mov rax, [rbp+ptr]
mov rax, [rax]
test rax, rax
jz short loc_436BD2

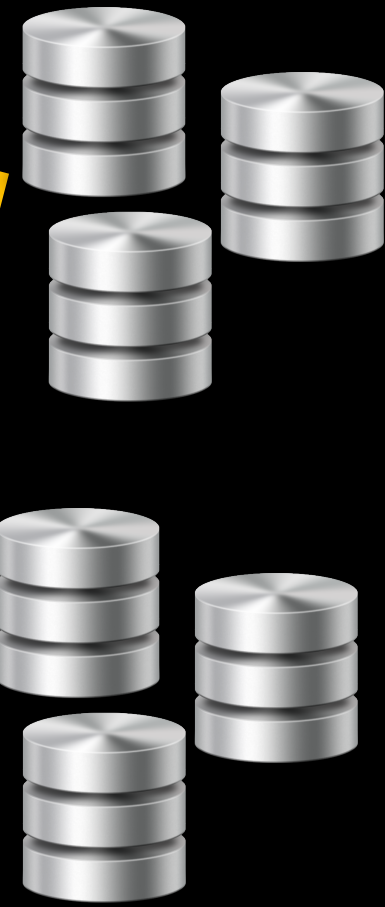
loc_436BD2:
nop
ca

loc_436BD9:
mov rax, [rbp+ptr]
mov rax, [rax]
mov rdi, rax
call sub_436DA9
```

67,246 Labeled Procedures

Strip & Obfuscate APIs

Strip



8:1:1 Package-Based Split

Evaluation Results

Nero-GNN

Nero-Transformer

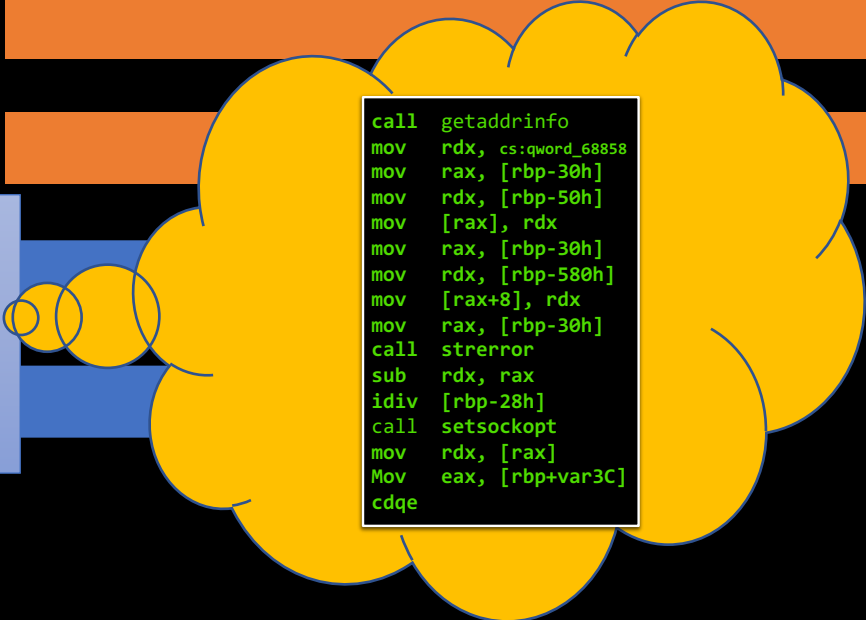
Nero-LSTM

DIRE [Lacomis et al. 2019]

Debin [He et al. 2018]

LSTM-text

Transformer-text



```
call  getaddrinfo
mov   rdx, cs:qword_68858
mov   rax, [rbp-30h]
mov   rdx, [rbp-50h]
mov   [rax], rdx
mov   rax, [rbp-30h]
mov   rdx, [rbp-580h]
mov   [rax+8], rdx
mov   rax, [rbp-30h]
call  strerror
sub   rdx, rax
idiv [rbp-28h]
call  setsockopt
mov   rdx, [rax]
Mov   eax, [rbp+var3C]
cdqe
```

Evaluation Results

Nero-GNN

Nero-Transformer

Nero-LSTM

DIRE [Lacomis et al. 2019]

Debin [He et al. 2018]

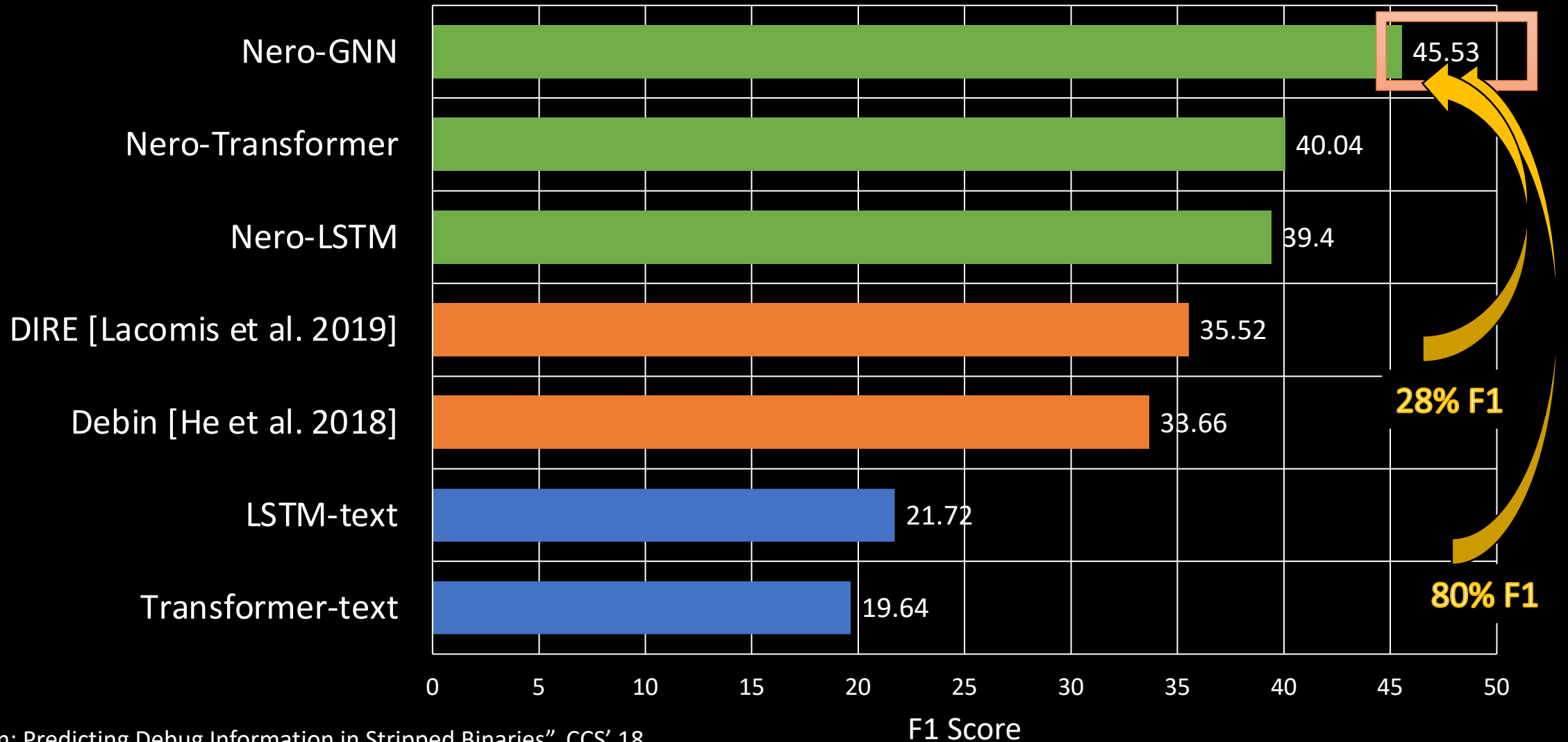
LSTM-text

Transformer-text

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

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

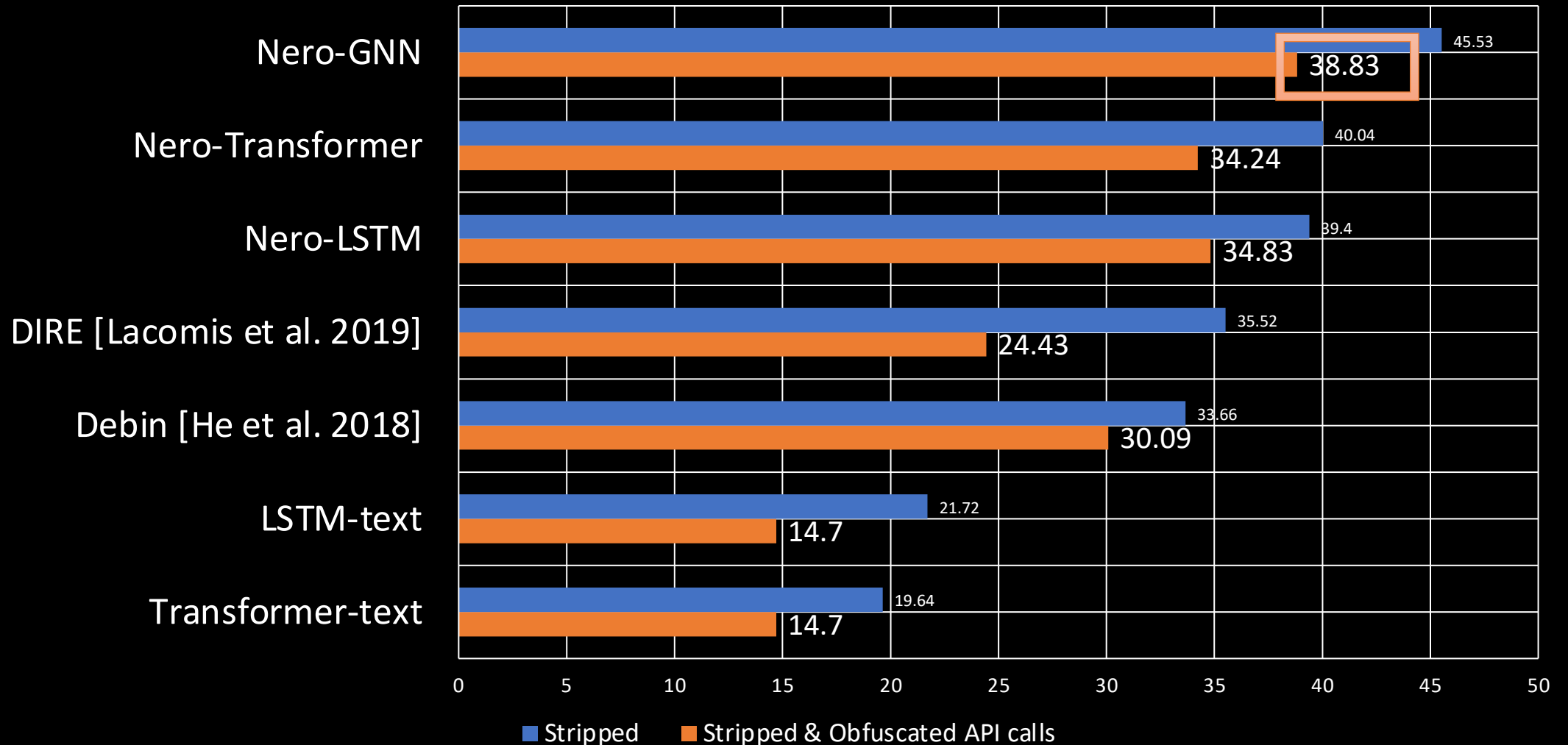
Evaluation Results



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

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

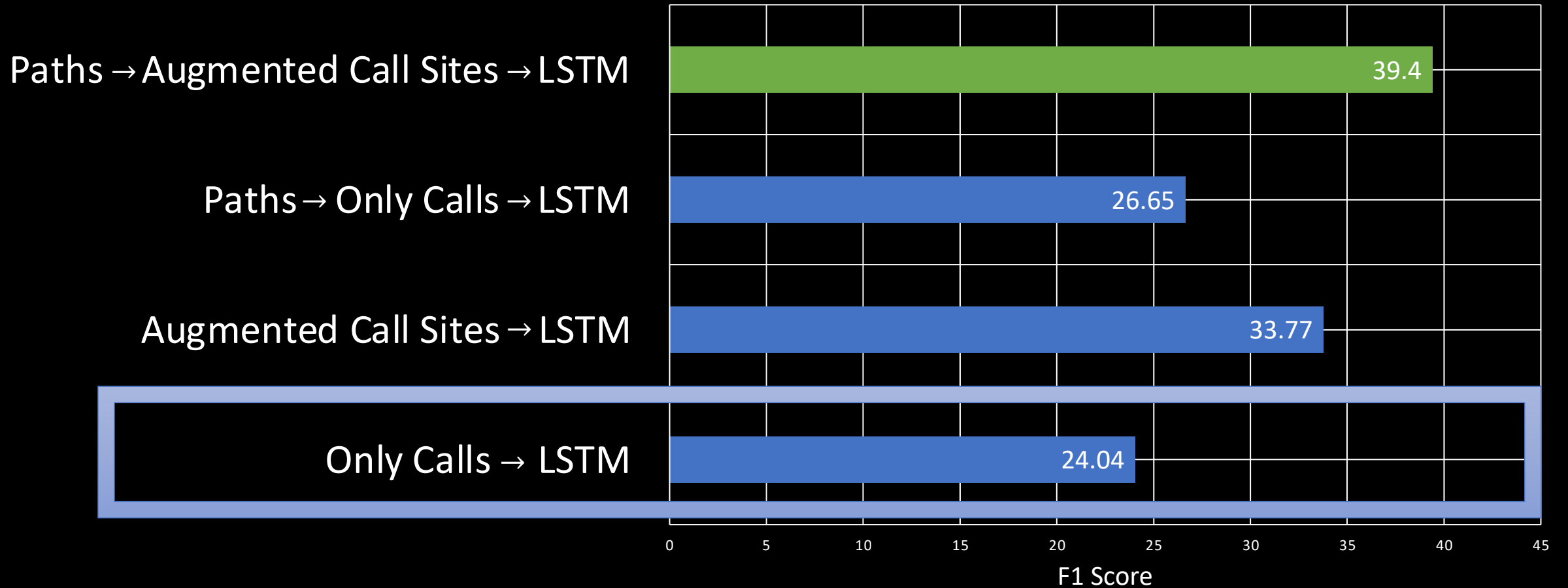
Evaluation Results



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

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

Ablation Study



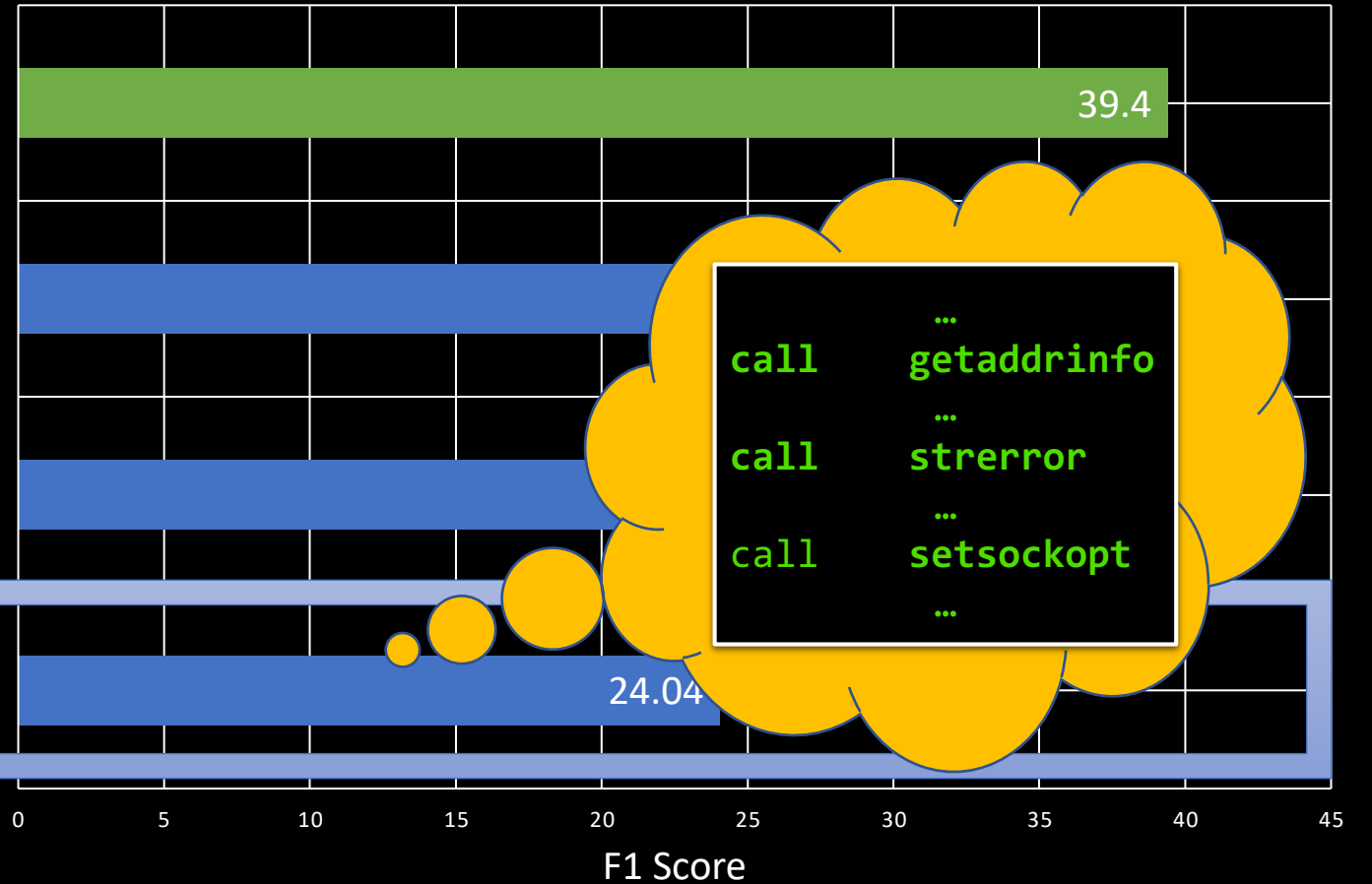
Ablation Study

Paths → Augmented Call Sites → LSTM

Paths → Only Calls → LSTM

Augmented Call Sites → LSTM

Only Calls → LSTM



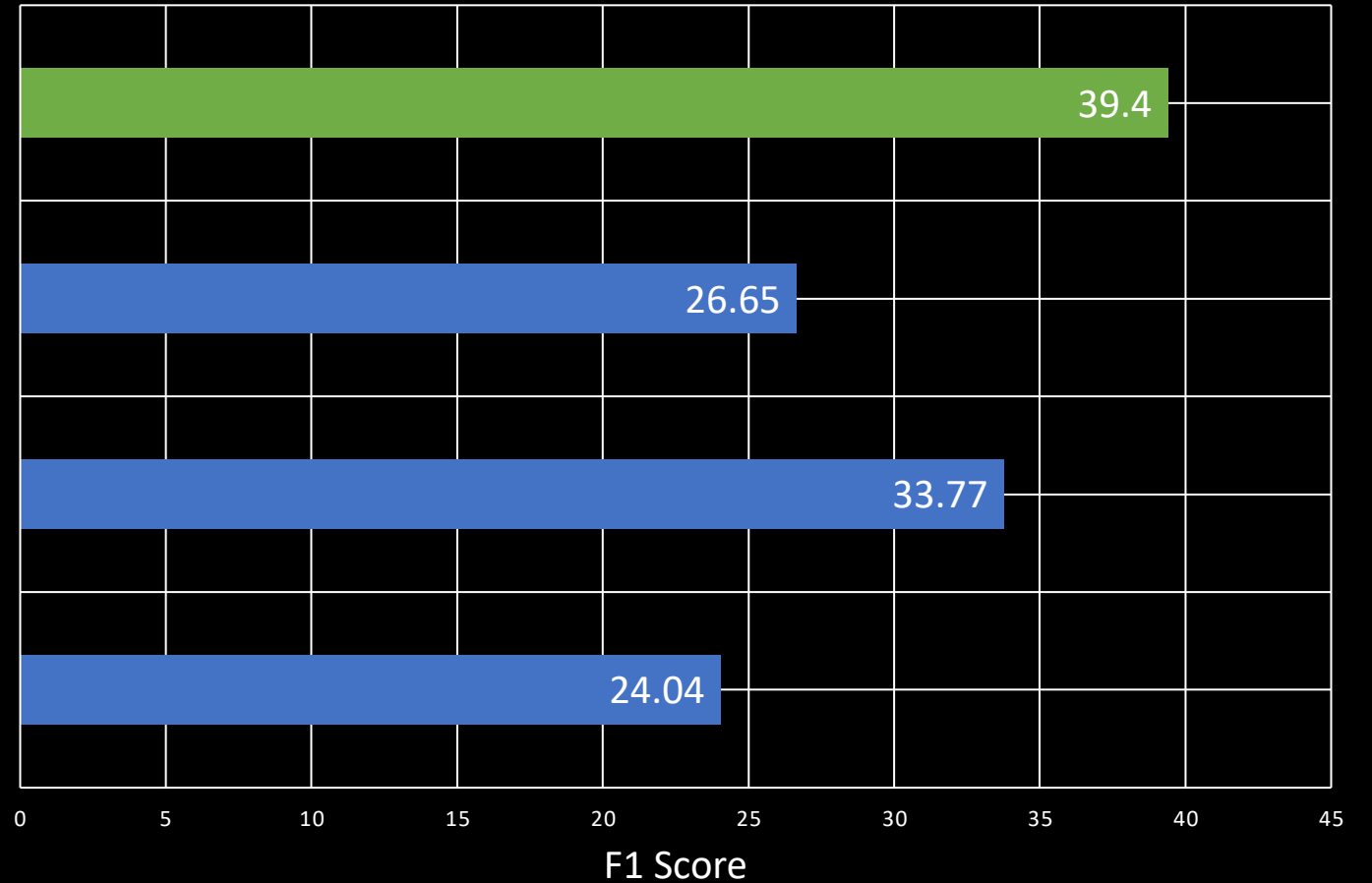
Ablation Study

Paths → Augmented Call Sites → LSTM

Paths → Only Calls → LSTM

Augmented Call Sites → LSTM

Only Calls → LSTM



Qualitative 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 |

Qualitative Evaluation

| Error Type | Ground Truth | Predicted Name |
|---------------------------------------|---------------------------|-----------------------------|
| Programmers Vs English Language | i18n_initialize | i18n_init |
| | split_cfg_path | split_cfg_path |
| | add_env_opt | add_env_opt |
| Date Structure Name Missing | get_best_speed | get_best_speed |
| | 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 |

Measured F1 is actually a lower-bound

Takeaway Messages

Augmented call sites serve as a strong basis for procedure representations

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

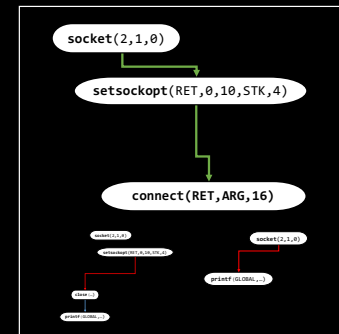
```
mov [rbp-50], rdi  
mov rdi, [rbp-50]  
mov rsi, rdi
```

`connect(rdi, rsi, rdx)`

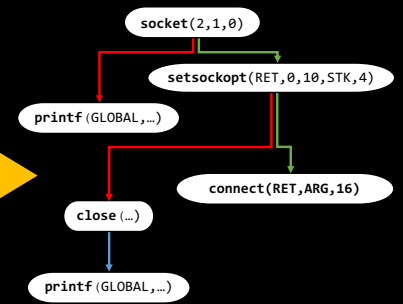
```
mov rdx, 16
```

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

Reconstructing the CFG enables the use of seq2seq and GNN models



LSTM & Transformer



GNN

