

Breaking State-of-the-Art Binary Code Obfuscation via Program Synthesis

Black Hat Asia, Singapore

March 22, 2018

Tim Blazytko, @mr_phrazer
<http://synthesis.to>

Moritz Contag, @dwuid
<https://dwuid.com>

Chair for Systems Security
Ruhr-Universität Bochum
<firstname.lastname>@rub.de



Syntia: Synthesizing the Semantics of Obfuscated Code

Tim Blazytko, Moritz Contag, Cornelius Aschermann,
and Thorsten Holz, *Ruhr-Universität Bochum*

<https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/blazytko>

This paper is included in the Proceedings of the
26th USENIX Security Symposium
August 16–18, 2017 • Vancouver, BC, Canada

ISBN 978-1-931971-40-9

Setting the Scene

- ➊ Obfuscated code, semantics?
- ➋ Traditional deobfuscation techniques
- ➔ Orthogonal approach

Motivation

Prevent Complicate reverse engineering attempts.

- Intellectual Property
- Malicious Payloads
- Digital Rights Management

Motivation

Prevent Complicate reverse engineering attempts.

- Intellectual Property
- Malicious Payloads
- Digital Rights Management

“We achieved our goals. We were uncracked for **13 whole days**.”

– Martin Slater, 2K Australia, on *BioShock* (2007).

How to protect software?

Approaches

Abuse shortcomings of file parsers and other tools of the trade.

- `fld tbyte ptr [__bad_values]` crashing OllyDbg 1.10.
- Fake `SizeOfImage` crashing process dumpers.

Approaches

Abuse shortcomings of file parsers and other tools of the trade.

- `fld tbyte ptr [__bad_values]` crashing OllyDbg 1.10.
- Fake `SizeOfImage` crashing process dumpers.

Detect artifacts of the debugging process.

- `PEB.BeingDebugged` bit being set.
- `int 2D` and exception handling in debuggers.

Approaches

A screenshot of a search results page from a search engine. The search query in the bar is "game does not start debugger detected". The results are categorized under "All" (highlighted in blue), with other categories like Videos, Shopping, Images, News, and More available. A red box highlights the search results count: "About 6.370.000 results (0,51 seconds)". Below the results, a link is shown: "When i run this game i get a debugger error message Debugger ...". The URL for this link is partially visible as <https://support.ubi.com/.../When-i-run-this-game-i-get-a-debugger-error-message-De...>. A dropdown arrow is shown next to the URL. The snippet of the page content below the link reads: "When i run this game i get the following error message : Debugger Detected - Please close it down and restart! Windows NT ... Our game will not run while this application is running in memory, to stop this from happening you will need to stop MDM.exe as a startup process. Do the following : Goto the "Start" button --> "Run".

Requirements

1. We want the technique to be *semantics-preserving*.

Preserve the observable behavior of the application.

Requirements

1. We want the technique to be *semantics-preserving*.
2. We want to avoid external dependencies, focus on code only.

Assume white-box attack scenario.

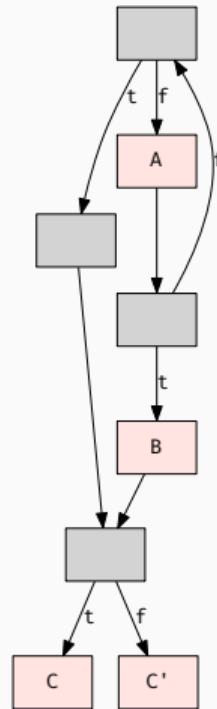
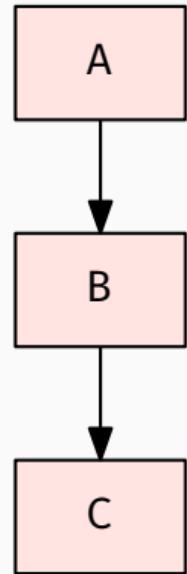
Requirements

1. We want the technique to be *semantics-preserving*.
2. We want to avoid external dependencies, focus on code only.
3. We want techniques where **effort(deploy) ≪ effort(attack)**.

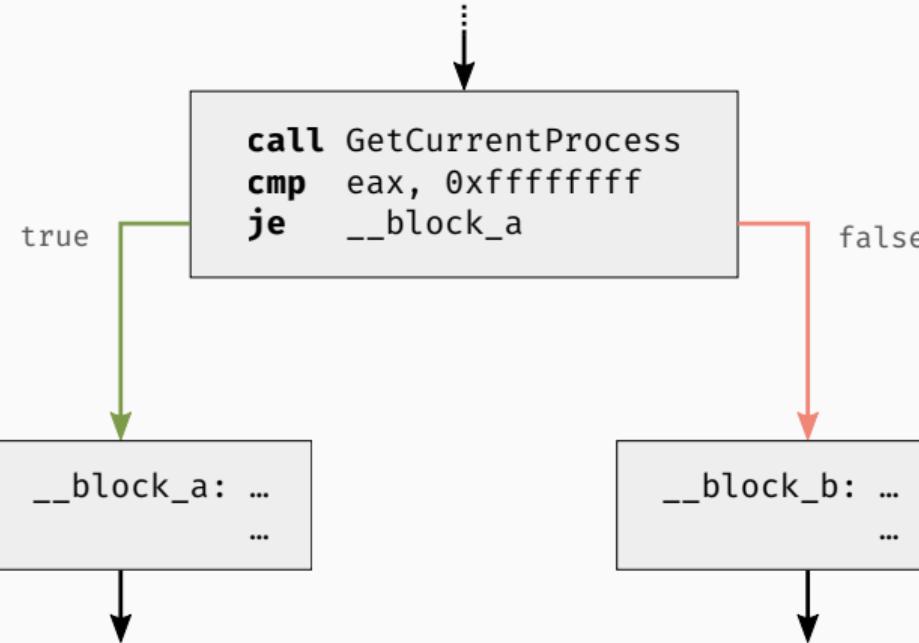
Anti-Debugging tricks are effort 1:1.

Code Obfuscation Techniques

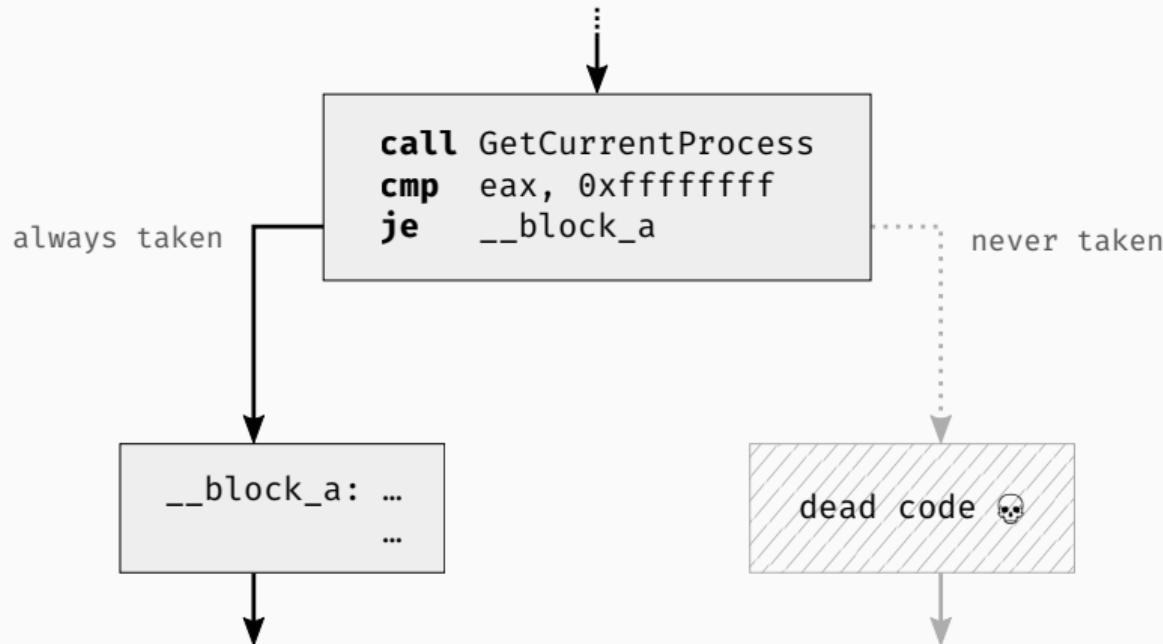
Opaque Predicates



Opaque Predicates

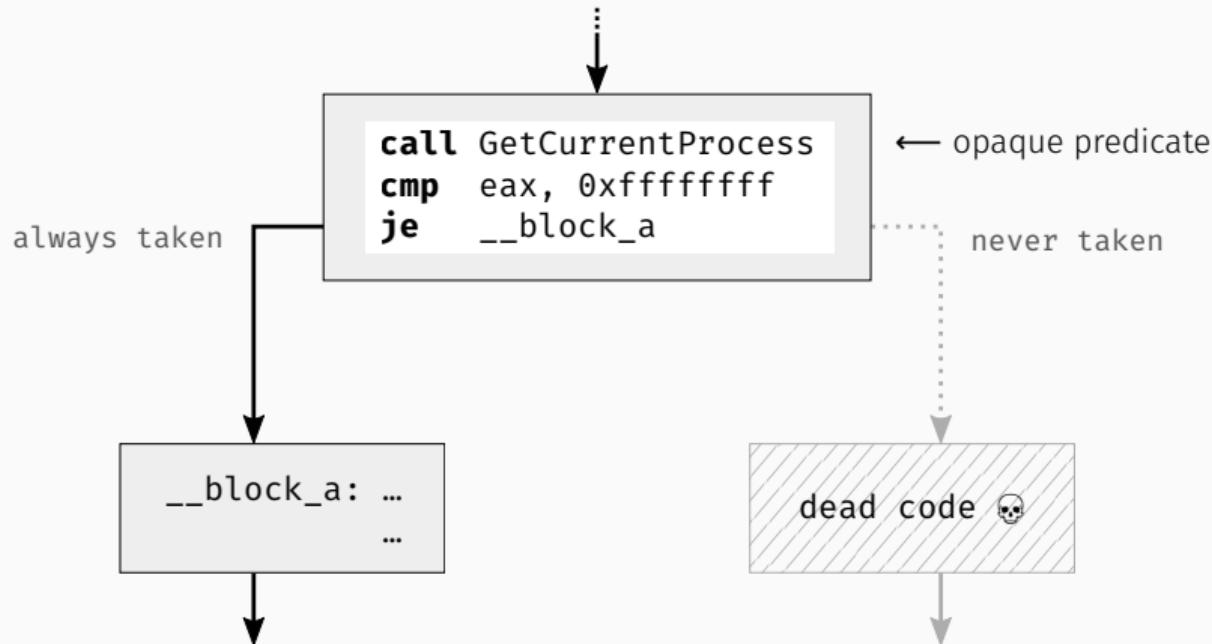


Opaque Predicates



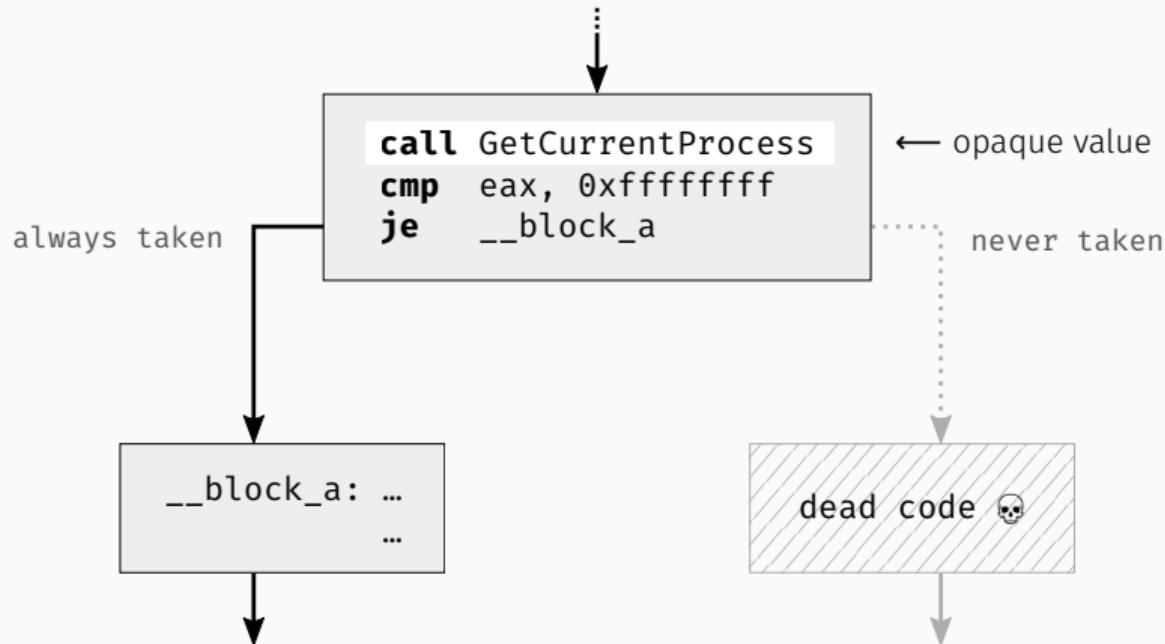
Opaque True Predicate

Opaque Predicates



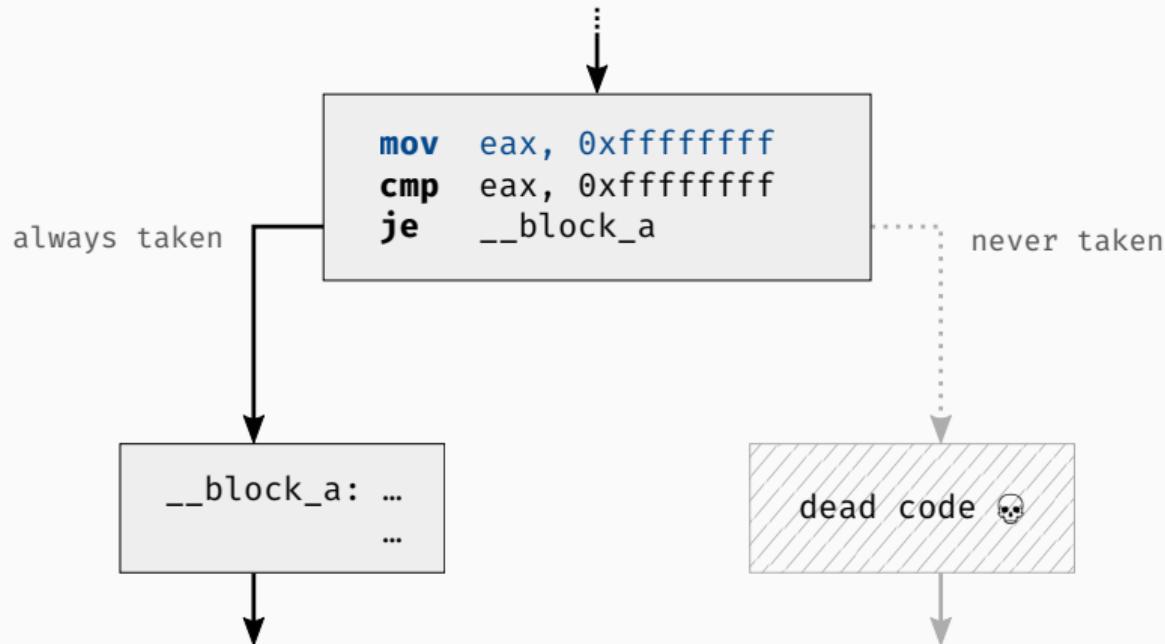
Opaque True Predicate

Opaque Predicates



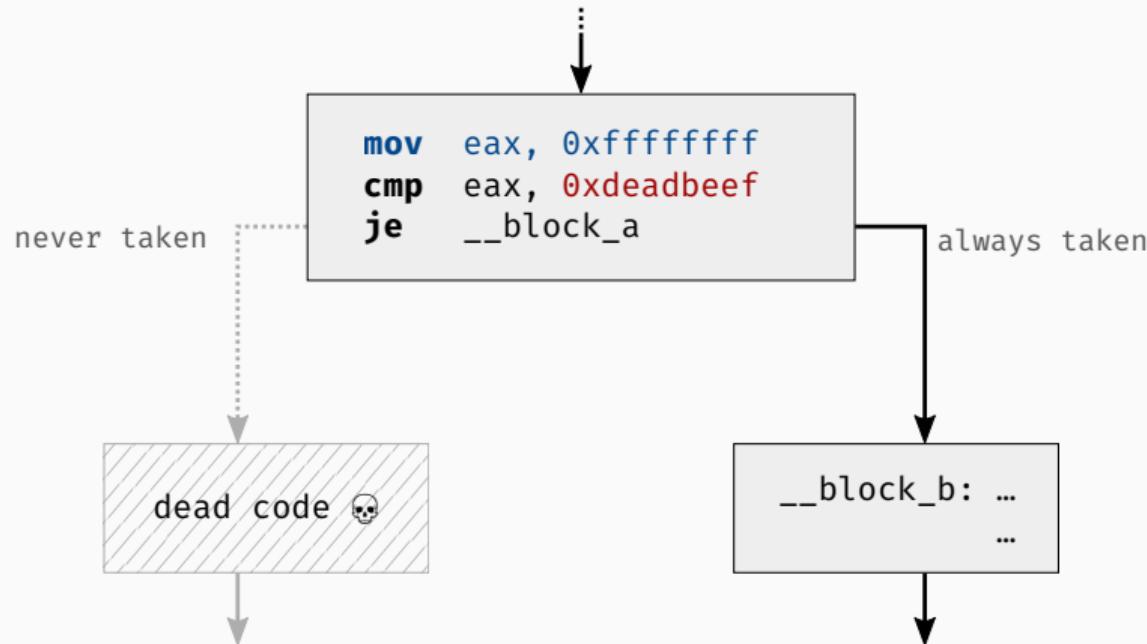
Opaque True Predicate

Opaque Predicates



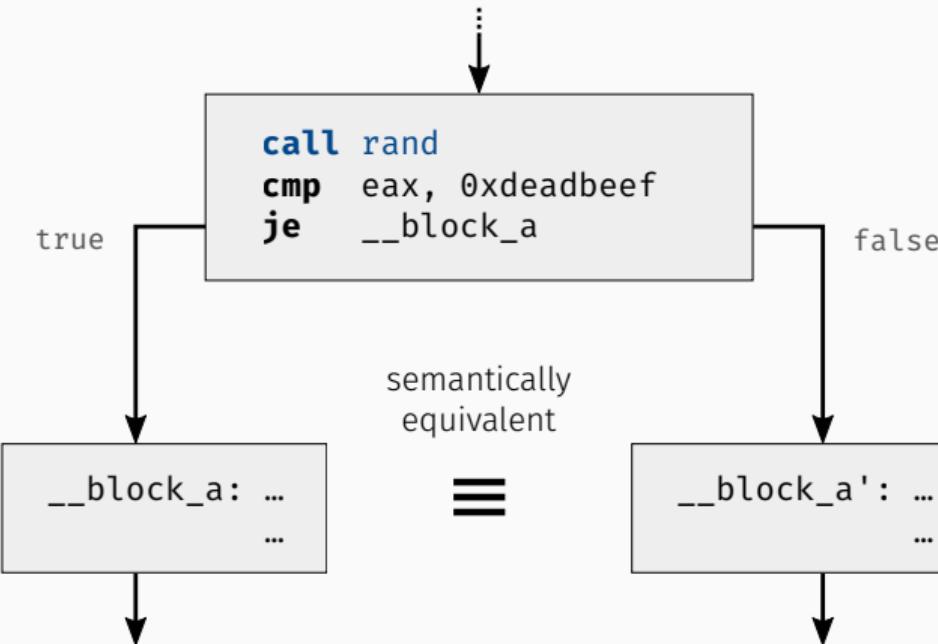
Opaque True Predicate

Opaque Predicates



Opaque False Predicate

Opaque Predicates



Random Opaque Predicate
duplicated block

Opaque Predicates

- ⊕ Increase in complexity (branch count, McCabe)
- ⊕ Can be built on hard problems (e.g., aliasing)
- ⊕ Forces analyst to encode additional knowledge
- ⊕ Hard to solve statically

⚠ Examples

- `GetCurrentProcess()` $\Rightarrow -1$
- `fldpi1` $\Rightarrow st(0) = \pi$
- $x^2 \geq 0 \quad \forall x$
- $x + 1 \neq x \quad \forall x$
- pointer A *must-alias* pointer B
- `checksum(code) = 0x1c43b5cf`

Opaque Predicates

- ⊕ Increase in complexity (branch count, McCabe)
- ⊕ Can be built on hard problems (e.g., aliasing)
- ⊕ Forces analyst to encode additional knowledge
- ⊕ Hard to solve statically
- ⊖ Solved for free using **concrete execution traces**

⚠ Examples

- `GetCurrentProcess()` $\Rightarrow -1$
- `fldpi1` $\Rightarrow st(0) = \pi$
- $x^2 \geq 0 \quad \forall x$
- $x + 1 \neq x \quad \forall x$
- pointer A *must-alias* pointer B
- $\text{checksum}(\text{code}) = 0x1c43b5cf$

Code Obfuscation Techniques

Virtual Machines

Virtual Machines

```
mov ecx, [esp+4]
xor eax, eax
mov ebx, 1

__secret_ip:
    mov edx, eax
    add edx, ebx
    mov eax, ebx
    mov ebx, edx
    loop __secret_ip

    mov eax, ebx
    ret
```

Virtual Machines

```
mov ecx, [esp+4]
xor eax, eax
mov ebx, 1

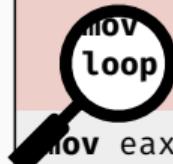
__secret_ip:
    mov edx, eax
    add edx, ebx
    mov eax, ebx
    mov ebx, edx
    loop __secret_ip

    mov eax, ebx
    ret
```

Virtual Machines

```
mov ecx, [esp+4]
xor eax, eax
mov ebx, 1

__secret_ip:
    mov edx, eax
    add edx, ebx
    mov eax, ebx
    mov ebx, edx
    mov loop __secret_ip
    mov eax, ebx
    ret
```

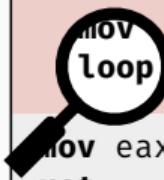


Virtual Machines

```
mov ecx, [esp+4]
xor eax, eax
mov ebx, 1
```

```
--secret_ip:
    mov edx, eax
    add edx, ebx
    mov eax, ebx
    mov ebx, edx
    loop --secret_ip
```

```
    mov eax, ebx
    ret
```



made-up instruction set

```
--bytecode: vld r1
            vld r0 vpop r2
            vpop r1 vldi #1
            vld r2 vld r3
            vld r1 vsub r3
            vadd r1 vld #0
            vld r2 veq r3
            vpop r0 vbr0 #-0E
```

Virtual Machines

```
mov ecx, [esp+4]
xor eax, eax
mov ebx, 1
```

```
--secret_ip:
push __bytecode
call vm_entry
```

```
mov eax, ebx
ret
```



made-up instruction set

```
--bytecode:
```

```
db 54 68 69 73 20 64 6f
db 65 73 6e 27 74 20 6c
db 6f 6f 6b 20 6c 69 6b
db 65 20 61 6e 79 74 68
db 69 6e 67 20 74 6f 20
db 6d 65 2e de ad be ef
```

Virtual Machines

```
mov ecx, [esp+4]  
xor eax, eax  
mov ebx, 1
```

```
--secret_ip:  
push __bytecode  
call vm_entry
```

```
mov eax, ebx  
ret
```



made-up instruction set

--bytecode:

```
db 54 68 69 73 20 64 6f  
db 65 73 6e 27 74 20 6c  
db 6f 6f 6b 20 6c 69 6b  
db 65 20 61 6e 79 74 68  
69 6e 67 20 74 6f 20  
65 2e de ad be ef
```



Virtual Machines

Core Components

VM Entry/Exit Context Switch: native context \Leftrightarrow virtual context

VM Dispatcher Fetch–Decode–Execute loop

Handler Table Individual VM ISA instruction semantics

- **Entry** Copy native context (registers, flags) to VM context.
- **Exit** Copy VM context back to native context.
- Mapping from native to virtual registers is often 1:1.

Virtual Machines

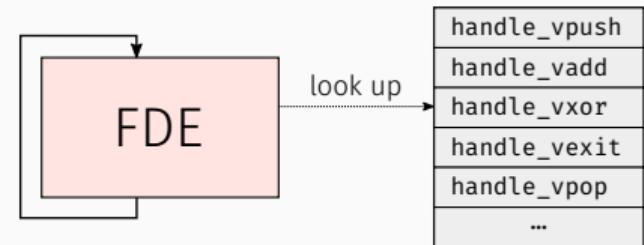
Core Components

VM Entry/Exit Context Switch: native context \Leftrightarrow virtual context

VM Dispatcher Fetch–Decode–Execute loop

Handler Table Individual VM ISA instruction semantics

1. Fetch and decode instruction
2. Forward virtual instruction pointer
3. Look up handler for opcode in handler table
4. Invoke handler



Virtual Machines

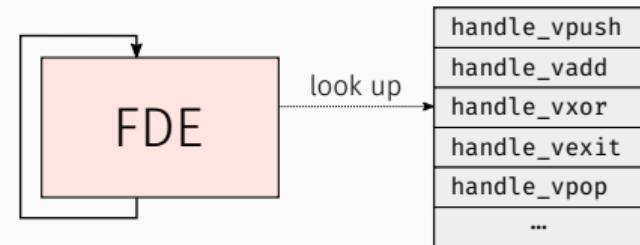
Core Components

VM Entry/Exit Context Switch: native context \Leftrightarrow virtual context

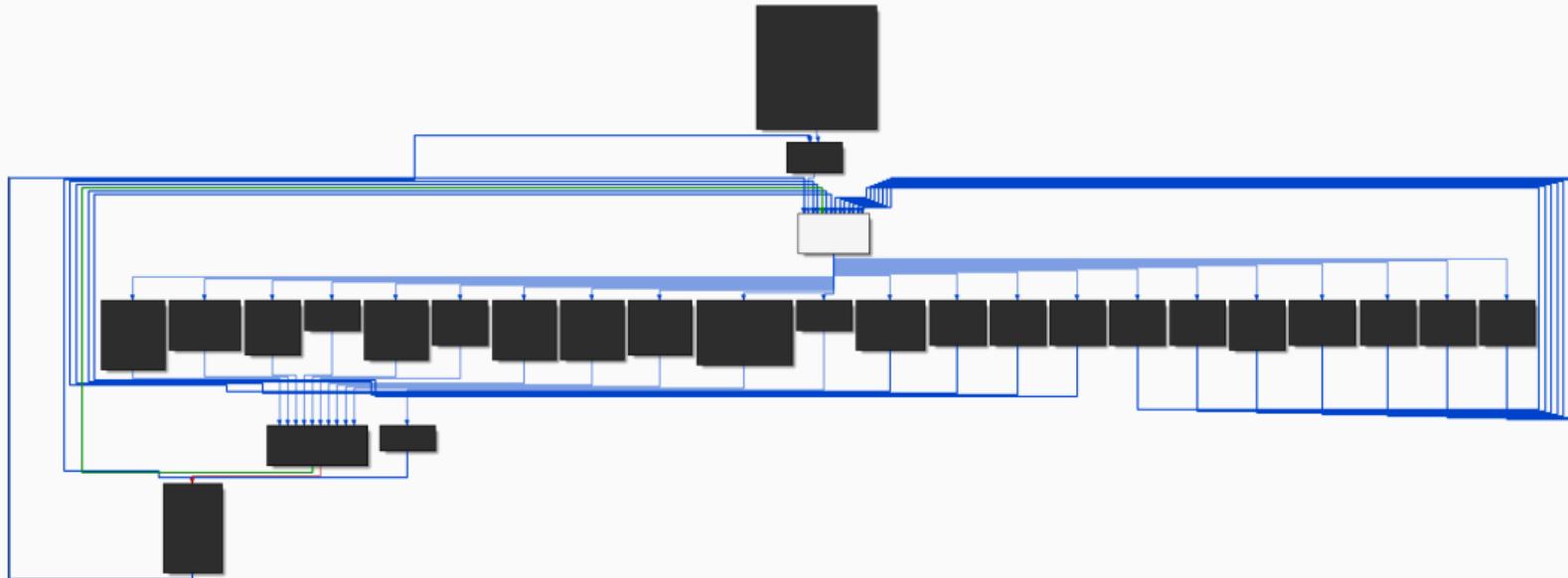
VM Dispatcher Fetch–Decode–Execute loop

Handler Table Individual VM ISA instruction semantics

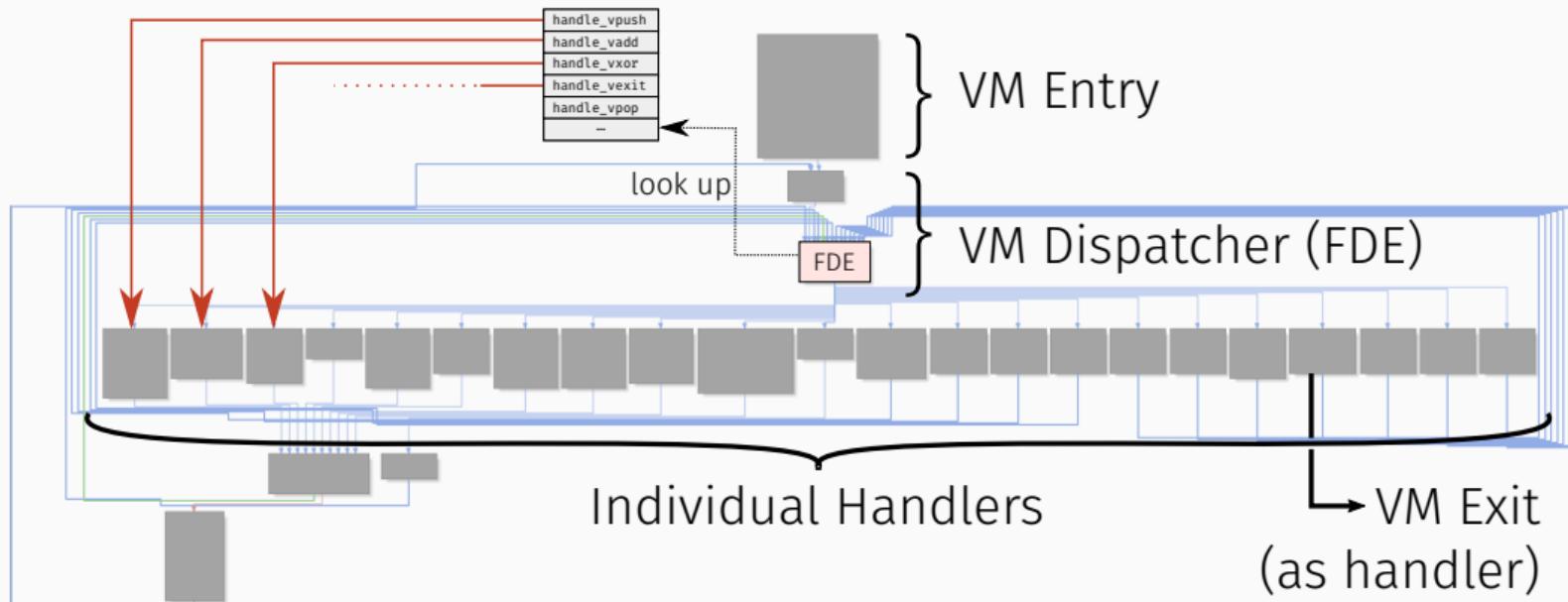
- Table of function pointers indexed by opcode
- One handler per virtual instruction
- Each handler decodes operands and updates VM context



Virtual Machines



Virtual Machines



Virtual Machines

```
__vm_dispatcher:  
    mov    bl, [rsi]  
    inc    rsi  
    movzx  rax, bl  
    jmp    __handler_table[rax * 8]
```

VM Dispatcher

rsi – virtual instruction pointer
rbp – VM context

Virtual Machines

```
__vm_dispatcher:  
    mov    bl, [rsi]  
    inc    rsi  
    movzx  rax, bl  
    jmp    __handler_table[rax * 8]
```

VM Dispatcher

rsi – virtual instruction pointer
rbp – VM context

```
--handle_vnor:  
    mov    rcx, [rbp]  
    mov    rbx, [rbp + 4]  
    not    rcx  
    not    rbx  
    and    rcx, rbx  
    mov    [rbp + 4], rcx  
    pushf  
    pop    [rbp]  
    jmp    __vm_dispatcher
```

Handler performing **nor**
(with flag side-effects)

Virtual Machine Hardening

Hardening Technique #1 – Obfuscating individual VM components.

- Handlers are *conceptually simple*.

Hardening Technique #1 – Obfuscating individual VM components.

- Handlers are *conceptually simple*.
- Apply traditional code obfuscation transformations:
 - Substitution (`mov rax, rbx` → `push rbx; pop rax`)
 - Opaque Predicates
 - Junk Code
 - ...

```
mov eax, dword [rbp]
mov ecx, dword [rbp+4]
cmp r11w, r13w
sub rbp, 4
not eax
clc
cmc
cmp rdx, 0x28b105fa
not ecx
cmp r12b, r9b
```

Hardening Technique #2 – Duplicating VM handlers.

- Handler table is typically indexed using one byte (= 256 entries).

Hardening Technique #2 – Duplicating VM handlers.

- Handler table is typically indexed using one byte (= 256 entries).
- **Idea:** *Duplicate* existing handlers to populate full table.
- Use traditional obfuscation techniques to impede *code similarity* analyses.

Goal: Increase workload of reverse engineer.

handle_vpush

handle_vadd

handle_vnor

handle_vpop

handle_vpush
handle_vadd
handle_vnor
handle_vpop



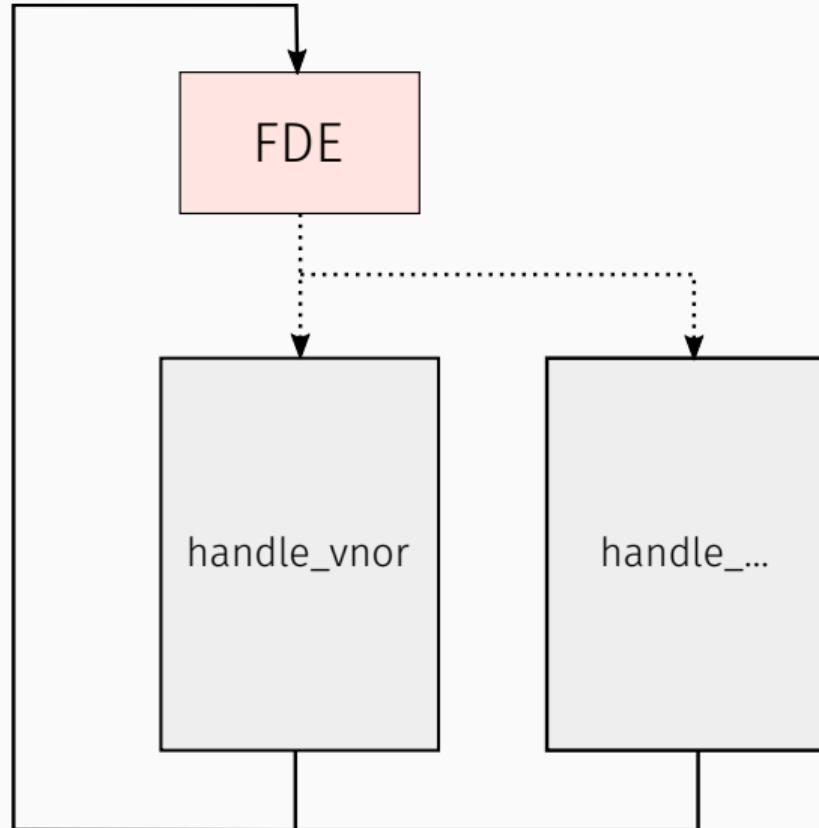
handle_vpush
handle_vadd
handle_vnor ''
handle_vpop
handle_vadd'
handle_vnor
handle_vnor '
handle_vadd ''

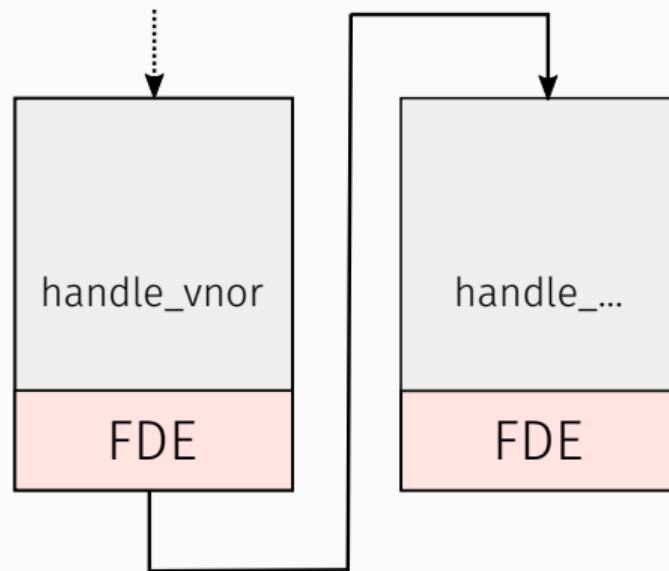
Hardening Technique #3 – No central VM dispatcher.

- A *central* VM dispatcher allows attacker to easily observe VM execution.
- **Idea:** Instead of branching to the central dispatcher, *inline* it into each handler.

Goal: No “single point of failure”.

(Themida, VMProtect Demo)





Threaded Code

James R. Bell
Digital Equipment Corporation

The concept of "threaded code" is presented as an alternative to machine language code. Hardware and software realizations of it are given. In software it is realized as interpretive code not needing an interpreter. Extensions and optimizations are mentioned.

Key Words and Phrases: interpreter, machine code, time tradeoff, space tradeoff, compiled code, subroutine calls, threaded code

CR Categories: 4.12, 4.13, 6.33

Fig. 2 Flow of control: interpretive code.

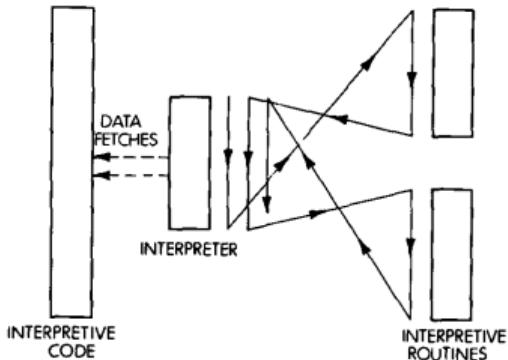
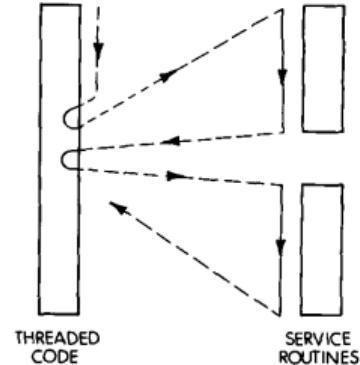


Fig. 3. Flow of control: threaded code.



Hardening Technique #4 – No explicit handler table.

- An *explicit* handler table easily reveals all VM handlers.

Hardening Technique #4 – No explicit handler table.

- An *explicit* handler table easily reveals all VM handlers.
- **Idea:** Instead of querying an explicit handler table,
encode the next handler address in the VM instruction itself.

Goal: Hide location of handlers that have not been executed yet.

(VMProtect Full, SolidShield)

Hardening Technique #4 – No explicit handler table.

- An *explicit* handler table easily reveals all VM handlers.
- Idea:  Instead of having an explicit handler table,
the VM instruction itself.

Goal: Hide location of handlers that have not been executed yet.

(VMProtect Full, SolidShield)

Hardening Technique #4 – No explicit handler table.

- An *explicit* handler table easily reveals all VM handlers.

- IdeaThe diagram illustrates a memory structure for a virtual machine instruction. It consists of four adjacent boxes. The first three boxes are green and contain the labels "opcode", "op 0", and "op 1" respectively. The fourth box is red and contains the label "next handler addr".

Goal: Hide location of handlers that have not been executed yet.

(VMProtect Full, SolidShield)

SOFTWARE-PRACTICE AND EXPERIENCE, VOL. 11, 963-973 (1981)

Interpretation Techniques^{*}

PAUL KLINT

Mathematical Centre, P.O. Box 4079, 1009AB Amsterdam, The Netherlands

SUMMARY

The relative merits of implementing high level programming languages by means of interpretation or compilation are discussed. The properties and the applicability of interpretation techniques known as classical interpretation, direct threaded code and indirect threaded code are described and compared.

KEY WORDS Interpretation versus compilation Interpretation techniques Instruction encoding Code generation Direct threaded code Indirect threaded code.

Hardening Technique #5 – Blinding VM bytecode.

- *Global analyses* on the bytecode possible, easy to patch instructions.

Hardening Technique #5 – Blinding VM bytecode.

- *Global analyses* on the bytecode possible, easy to patch instructions.
- Idea:
 - Flow-sensitive instruction decoding (“decryption” based on key register).
 - Custom decryption routine per handler, diversification.
 - Patching requires re-encryption of subsequent bytecode.

Goal: Hinder global analyses of bytecode and patching.

operand $\leftarrow [vIP + 0]$

context $\leftarrow \text{semantics}(\text{context}, \text{operand})$
next_handler $\leftarrow [vIP + 4]$

$vIP \leftarrow vIP + 8$

jmp *next_handler*

operand

$\leftarrow [vIP + 0]$

 *operand*

$\leftarrow \text{unmangle}(\textit{operand}, \textbf{key})$

 **key**

$\leftarrow \text{unmangle}'(\textbf{key}, \textit{operand})$

context

$\leftarrow \text{semantics}(\textit{context}, \textit{operand})$

next_handler

$\leftarrow [vIP + 4]$

 *next_handler*

$\leftarrow \text{unmangle}''(\textit{next_handler}, \textbf{key})$

 **key**

$\leftarrow \text{unmangle}'''(\textbf{key}, \textit{next_handler})$

$vIP \leftarrow vIP + 8$

jmp *next_handler*

Code Obfuscation Techniques

Mixed Boolean-Arithmetic

Mixed Boolean-Arithmetic

What does this expression compute?

$$(x \oplus y) + 2 \cdot (x \wedge y)$$

Mixed Boolean-Arithmetic

What does this expression compute?

$$\begin{aligned}(x \oplus y) + 2 \cdot (x \wedge y) \\ = x + y\end{aligned}$$

Mixed Boolean-Arithmetic

What does this expression compute?

$$(((x \oplus y) + ((x \wedge y) \ll 1)) \vee z) + (((x \oplus y) + ((x \wedge y) \ll 1)) \wedge z)$$

Mixed Boolean-Arithmetic

What does this expression compute?

$$\begin{aligned} & (((x \oplus y) + ((x \wedge y) \ll 1)) \vee z) + (((x \oplus y) + ((x \wedge y) \ll 1)) \wedge z) \\ &= x + y + z \end{aligned}$$

- Boolean identities?
- Arithmetic identities?
- Karnaugh-Veitch maps?

$$A \cdot 0 = 0$$

$$A + B = \overline{\overline{A} \cdot \overline{B}}$$

$$x^2 - y^2 = (x + y)(x - y)$$

		AB		CD
		00	01	
00	00	0	0	1
	01	0	0	1
01	00	0	0	1
	01	1	1	1
10	00	0	1	1
	11	1	1	1

Mixed Boolean-Arithmetic

Boolean-arithmetic algebra $\text{BA}[n]$

$(B^n, \wedge, \vee, \oplus, \neg, \leq, \geq, >, <, \leq^s, \geq^s, >^s, <^s, \neq, =, \gg^s, \gg, \ll, +, -, \cdot)$
is a Boolean-arithmetic algebra $\text{BA}[n]$, for $n > 0$, $B = \{0, 1\}$.

$\text{BA}[n]$ includes, amongst others, both:

- Boolean algebra $(B^n, \wedge, \vee, \neg),$
- Integer modular ring $\mathbb{Z}/(2^n).$

No techniques to simplify
such expressions easily!

Deobfuscation

Symbolic Execution

```
--handle_vnor:  
    mov    rcx, [rbp]  
    mov    rbx, [rbp + 4]  
    not    rcx  
    not    rbx  
    and    rcx, rbx  
    mov    [rbp + 4], rcx  
    pushf  
    pop    [rbp]  
    jmp    __vm_dispatcher
```

Handler performing **nor**
(with flag side-effects)

Symbolic Execution

```
__handle_vnor:  
• mov rcx, [rbp]  
  mov rbx, [rbp + 4]  
  not rcx  
  not rbx  
  and rcx, rbx  
  mov [rbp + 4], rcx  
  pushf  
  pop [rbp]  
  jmp __vm_dispatcher
```

rcx \leftarrow [rbp]

Handler performing **nor**
(with flag side-effects)

Symbolic Execution

```
__handle_vnor:  
    mov    rcx, [rbp]  
• mov    rbx, [rbp + 4]  
    not    rcx  
    not    rbx  
    and    rcx, rbx  
    mov    [rbp + 4], rcx  
    pushf  
    pop    [rbp]  
    jmp    __vm_dispatcher
```

rcx \leftarrow [rbp]
rbx \leftarrow [rbp + 4]

Handler performing **nor**
(with flag side-effects)

Symbolic Execution

```
__handle_vnor:  
    mov    rcx, [rbp]  
    mov    rbx, [rbp + 4]  
• not   rcx  
    not   rbx  
    and   rcx, rbx  
    mov    [rbp + 4], rcx  
    pushf  
    pop    [rbp]  
    jmp    __vm_dispatcher
```

rcx \leftarrow [rbp]
rbx \leftarrow [rbp + 4]
rcx $\leftarrow \neg$ **rcx** = \neg [rbp]

Handler performing **nor**
(with flag side-effects)

Symbolic Execution

```
__handle_vnor:  
    mov    rcx, [rbp]  
    mov    rbx, [rbp + 4]  
    not    rcx  
• not    rbx  
    and    rcx, rbx  
    mov    [rbp + 4], rcx  
    pushf  
    pop    [rbp]  
    jmp    __vm_dispatcher
```

```
rcx ← [rbp]  
rbx ← [rbp + 4]  
rcx ←  $\neg$ rcx =  $\neg$ [rbp]  
rbx ←  $\neg$ rbx =  $\neg$ [rbp + 4]
```

Handler performing **nor**
(with flag side-effects)

Symbolic Execution

```
__handle_vnor:  
    mov    rcx, [rbp]  
    mov    rbx, [rbp + 4]  
    not    rcx  
    not    rbx  
    • and   rcx, rbx  
    mov    [rbp + 4], rcx  
    pushf  
    pop    [rbp]  
    jmp    __vm_dispatcher
```

$$\begin{aligned} \text{rcx} &\leftarrow [\text{rbp}] \\ \text{rbx} &\leftarrow [\text{rbp} + 4] \\ \text{rcx} &\leftarrow \neg \text{rcx} = \neg [\text{rbp}] \\ \text{rbx} &\leftarrow \neg \text{rbx} = \neg [\text{rbp} + 4] \\ \text{rcx} &\leftarrow \text{rcx} \wedge \text{rbx} \\ &= (\neg [\text{rbp}]) \wedge (\neg [\text{rbp} + 4]) \end{aligned}$$

Handler performing **nor**
(with flag side-effects)

Symbolic Execution

```
__handle_vnor:  
    mov    rcx, [rbp]  
    mov    rbx, [rbp + 4]  
    not    rcx  
    not    rbx  
    • and   rcx, rbx  
    mov    [rbp + 4], rcx  
    pushf  
    pop    [rbp]  
    jmp    __vm_dispatcher
```

$$\begin{aligned} \text{rcx} &\leftarrow [\text{rbp}] \\ \text{rbx} &\leftarrow [\text{rbp} + 4] \\ \text{rcx} &\leftarrow \neg \text{rcx} = \neg [\text{rbp}] \\ \text{rbx} &\leftarrow \neg \text{rbx} = \neg [\text{rbp} + 4] \\ \text{rcx} &\leftarrow \text{rcx} \wedge \text{rbx} \\ &= (\neg [\text{rbp}]) \wedge (\neg [\text{rbp} + 4]) \\ &= [\text{rbp}] \downarrow [\text{rbp} + 4] \end{aligned}$$

Handler performing **nor**
(with flag side-effects)

Symbolic Execution

```
__handle_vnor:  
    mov    rcx, [rbp]  
    mov    rbx, [rbp + 4]  
    not    rcx  
    not    rbx  
    and    rcx, rbx  
•   mov    [rbp + 4], rcx  
    pushf  
    pop    [rbp]  
    jmp    __vm_dispatcher
```

$$\begin{aligned} \text{rcx} &\leftarrow [\text{rbp}] \\ \text{rbx} &\leftarrow [\text{rbp} + 4] \\ \text{rcx} &\leftarrow \neg \text{rcx} = \neg [\text{rbp}] \\ \text{rbx} &\leftarrow \neg \text{rbx} = \neg [\text{rbp} + 4] \\ \text{rcx} &\leftarrow \text{rcx} \wedge \text{rbx} \\ &= (\neg [\text{rbp}]) \wedge (\neg [\text{rbp} + 4]) \\ &= [\text{rbp}] \downarrow [\text{rbp} + 4] \\ [\text{rbp} + 4] &\leftarrow \text{rcx} = [\text{rbp}] \downarrow [\text{rbp} + 4] \end{aligned}$$

Handler performing **nor**
(with flag side-effects)

Symbolic Execution

```
__handle_vnor:  
    mov    rcx, [rbp]  
    mov    rbx, [rbp + 4]  
    not    rcx  
    not    rbx  
    and    rcx, rbx  
    mov    [rbp + 4], rcx  
• pushf  
    pop    [rbp]  
    jmp    __vm_dispatcher
```

$$\begin{aligned} \text{rcx} &\leftarrow [\text{rbp}] \\ \text{rbx} &\leftarrow [\text{rbp} + 4] \\ \text{rcx} &\leftarrow \neg \text{rcx} = \neg [\text{rbp}] \\ \text{rbx} &\leftarrow \neg \text{rbx} = \neg [\text{rbp} + 4] \\ \text{rcx} &\leftarrow \text{rcx} \wedge \text{rbx} \\ &= (\neg [\text{rbp}]) \wedge (\neg [\text{rbp} + 4]) \\ &= [\text{rbp}] \downarrow [\text{rbp} + 4] \\ [\text{rbp} + 4] &\leftarrow \text{rcx} = [\text{rbp}] \downarrow [\text{rbp} + 4] \\ \\ \text{rsp} &\leftarrow \text{rsp} - 4 \\ [\text{rsp}] &\leftarrow \text{flags} \end{aligned}$$

Handler performing **nor**
(with flag side-effects)

Symbolic Execution

```
__handle_vnor:  
    mov    rcx, [rbp]  
    mov    rbx, [rbp + 4]  
    not    rcx  
    not    rbx  
    and    rcx, rbx  
    mov    [rbp + 4], rcx  
    pushf  
• pop    [rbp]  
jmp    __vm_dispatcher
```

Handler performing **nor**
(with flag side-effects)

$$\begin{aligned} \text{rcx} &\leftarrow [\text{rbp}] \\ \text{rbx} &\leftarrow [\text{rbp} + 4] \\ \text{rcx} &\leftarrow \neg \text{rcx} = \neg [\text{rbp}] \\ \text{rbx} &\leftarrow \neg \text{rbx} = \neg [\text{rbp} + 4] \\ \text{rcx} &\leftarrow \text{rcx} \wedge \text{rbx} \\ &= (\neg [\text{rbp}]) \wedge (\neg [\text{rbp} + 4]) \\ &= [\text{rbp}] \downarrow [\text{rbp} + 4] \\ [\text{rbp} + 4] &\leftarrow \text{rcx} = [\text{rbp}] \downarrow [\text{rbp} + 4] \\ \\ \text{rsp} &\leftarrow \text{rsp} - 4 \\ [\text{rsp}] &\leftarrow \text{flags} \\ [\text{rbp}] &\leftarrow [\text{rsp}] = \text{flags} \\ \text{rsp} &\leftarrow \text{rsp} + 4 \end{aligned}$$

Symbolic Execution

```
--handle_vnor:  
    mov    rcx, [rbp]  
    mov    rbx, [rbp + 4]  
    not    rcx  
    not    rbx  
    and    rcx, rbx  
    mov    [rbp + 4], rcx  
    pushf  
    pop    [rbp]  
• jmp    __vm_dispatcher
```

Handler performing **nor**
(with flag side-effects)

$$\begin{aligned} \text{rcx} &\leftarrow [\text{rbp}] \\ \text{rbx} &\leftarrow [\text{rbp} + 4] \\ \text{rcx} &\leftarrow \neg \text{rcx} = \neg [\text{rbp}] \\ \text{rbx} &\leftarrow \neg \text{rbx} = \neg [\text{rbp} + 4] \\ & \vdots \\ & \boxed{[\text{rbp} + 4] \leftarrow ([\text{rbp}] \downarrow [\text{rbp} + 4])} \\ & \quad = [\text{rbp}] \downarrow [\text{rbp} + 4] \\ & \quad \vdots \\ & \quad [\text{rbp} + 4] \leftarrow \text{rcx} = [\text{rbp}] \downarrow [\text{rbp} + 4] \\ \text{rsp} &\leftarrow \text{rsp} - 4 \\ [\text{rsp}] &\leftarrow \text{flags} \\ [\text{rbp}] &\leftarrow [\text{rsp}] = \text{flags} \\ \text{rsp} &\leftarrow \text{rsp} + 4 \end{aligned}$$

Virtual Machine Handler

mov	eax, dword [rbp]	jmp	0xfffffffffffff63380
mov	ecx, dword [rbp + 4]	dec	eax
cmp	r11w, r13w	stc	
sub	rbp, 4	ror	eax, 1
not	eax	jmp	0xfffffffffffff2a70
clc		dec	eax
cmc		clc	
cmp	rdx, 0x28b105fa	bswap	eax
not	ecx	test	bp, 0x5124
cmp	r12b, r9b	neg	eax
cmc		test	dil, 0xe9
and	eax, ecx	cmp	bx, r14w
jmp	0xc239	cmc	
mov	word [rbp + 8], eax	push	rbx
pushfq		sub	bx, 0x49f8
movzx	eax, r10w	xor	dword [rsp], eax
and	ax, di	and	bh, 0xaf
pop	qword [rbp]	pop	rbx
sub	rsi, 4	movsx	rax, eax
shld	rax, rdx, 0x1b	test	r13b, 0x94
xor	ah, 0x4d	add	rdi, rax
mov	eax, dword [rsi]	jmp	0xfffffffffffffc67c7
cmp	ecx, r11d	lea	rax, [rsp + 0x140]
test	r10, 0x179708d5	cmp	rbp, rax
xor	eax, ebx	ja	0x6557b
		jmp	rdi

Mixed Boolean-Arithmetic Expression

```
int mixed_boolean(int A, int B, int C) {
    int result;

    result = (((1438524315 + (((1438524315 + C) + 1438524315 * ((2956783114 - -1478456685 * C) |
        (-1478456685 * (1668620215 - A) - 2956783115))) + A) - 1553572265)) + 1438524315 * ((2956783114 -
        -1478456685 * (((1438524315 + C) + 1438524315 * ((2956783114 - -1478456685 * C) | (-1478456685 *
            (1668620215 - A) - 2956783115))) + A) - 1553572265)) | (-1478456685 * (1668620215 - B) -
        2956783115)) - ((1438524315 + (1668620215 - (((1438524315 + C) + 1438524315 * ((2956783114 -
            -1478456685 * C) | (-1478456685 * (1668620215 - A) - 2956783115))) + A) - 1553572265)) +
        1438524315 * ((2956783114 - -1478456685 * (1668620215 - (((1438524315 + C) + 1438524315 *
            (2956783114 - -1478456685 * C) | (-1478456685 * (1668620215 - A) - 2956783115))) + A) -
            1553572265)) | (-1478456685 * B - 2956783115))) + 1553572265;

    return -1478456685 * result - 2956783115;
}
```


Symbolic Execution

- ⊕ Captures full semantics of executed code
- ⊕ Computer algebra system, some degree of simplification
- ⊖ Usability decreases with increasing *syntactic* complexity
 - Artificial complexity (substitution, ...)
 - Algebraic complexity (MBA)

Symbolic Execution

- ⊕ Captures full semantics of executed code
- ⊕ Computer algebra system, some degree of simplification
- ⊖ Usability decreases with increasing *syntactic* complexity
 - Artificial complexity (substitution, ...)
 - Algebraic complexity (MBA)

What if we could reason about *semantics* only instead of *syntax*?

Program Synthesis

Program Synthesis: A Semantic Approach

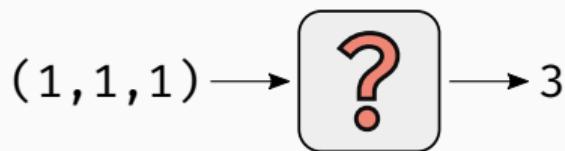
We use f as a black-box:

$$f(x, y, z) := (((x \oplus y) + ((x \wedge y) \cdot 2)) \vee z) + (((x \oplus y) + ((x \wedge y) \cdot 2)) \wedge z)$$

Program Synthesis: A Semantic Approach

We use f as a black-box:

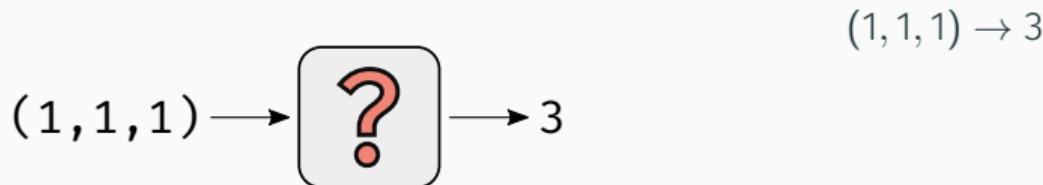
$$f(x, y, z) := (((x \oplus y) + ((x \wedge y) \cdot 2)) \vee z) + (((x \oplus y) + ((x \wedge y) \cdot 2)) \wedge z)$$



Program Synthesis: A Semantic Approach

We use f as a black-box:

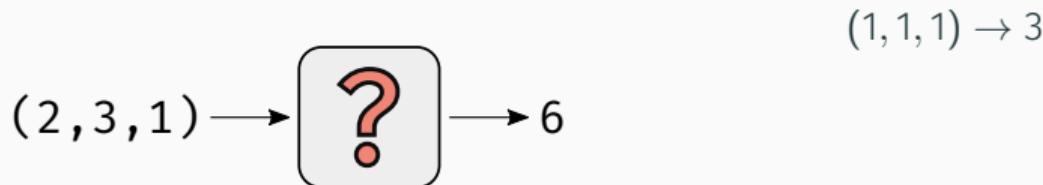
$$f(x, y, z) := (((x \oplus y) + ((x \wedge y) \cdot 2)) \vee z) + (((x \oplus y) + ((x \wedge y) \cdot 2)) \wedge z)$$



Program Synthesis: A Semantic Approach

We use f as a black-box:

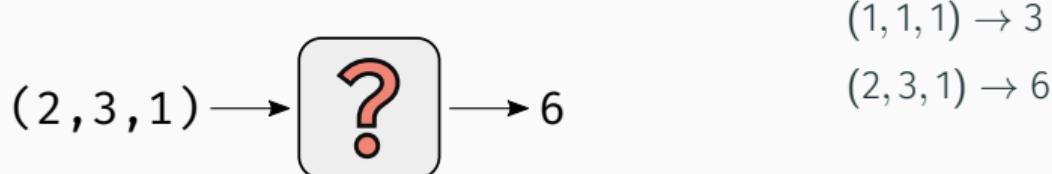
$$f(x, y, z) := (((x \oplus y) + ((x \wedge y) \cdot 2)) \vee z) + (((x \oplus y) + ((x \wedge y) \cdot 2)) \wedge z)$$



Program Synthesis: A Semantic Approach

We use f as a black-box:

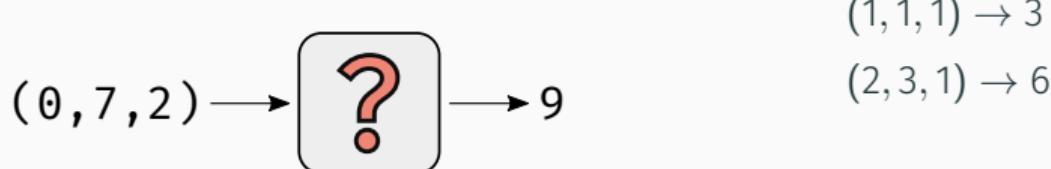
$$f(x, y, z) := (((x \oplus y) + ((x \wedge y) \cdot 2)) \vee z) + (((x \oplus y) + ((x \wedge y) \cdot 2)) \wedge z)$$



Program Synthesis: A Semantic Approach

We use f as a black-box:

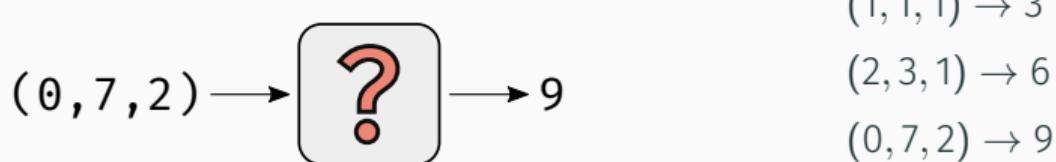
$$f(x, y, z) := (((x \oplus y) + ((x \wedge y) \cdot 2)) \vee z) + (((x \oplus y) + ((x \wedge y) \cdot 2)) \wedge z)$$



Program Synthesis: A Semantic Approach

We use f as a black-box:

$$f(x, y, z) := (((x \oplus y) + ((x \wedge y) \cdot 2)) \vee z) + (((x \oplus y) + ((x \wedge y) \cdot 2)) \wedge z)$$



Program Synthesis: A Semantic Approach

We use f as a black-box:

$$f(x, y, z) := (((x \oplus y) + ((x \wedge y) \cdot 2)) \vee z) + (((x \oplus y) + ((x \wedge y) \cdot 2)) \wedge z)$$

$$(1, 1, 1) \rightarrow 3$$

$$(2, 3, 1) \rightarrow 6$$

$$(0, 7, 2) \rightarrow 9$$

We **learn** a function that has the same I/O behavior:

Program Synthesis: A Semantic Approach

We use f as a black-box:

$$f(x, y, z) := (((x \oplus y) + ((x \wedge y) \cdot 2)) \vee z) + (((x \oplus y) + ((x \wedge y) \cdot 2)) \wedge z)$$

$$(1, 1, 1) \rightarrow 3$$

$$(2, 3, 1) \rightarrow 6$$

$$(0, 7, 2) \rightarrow 9$$

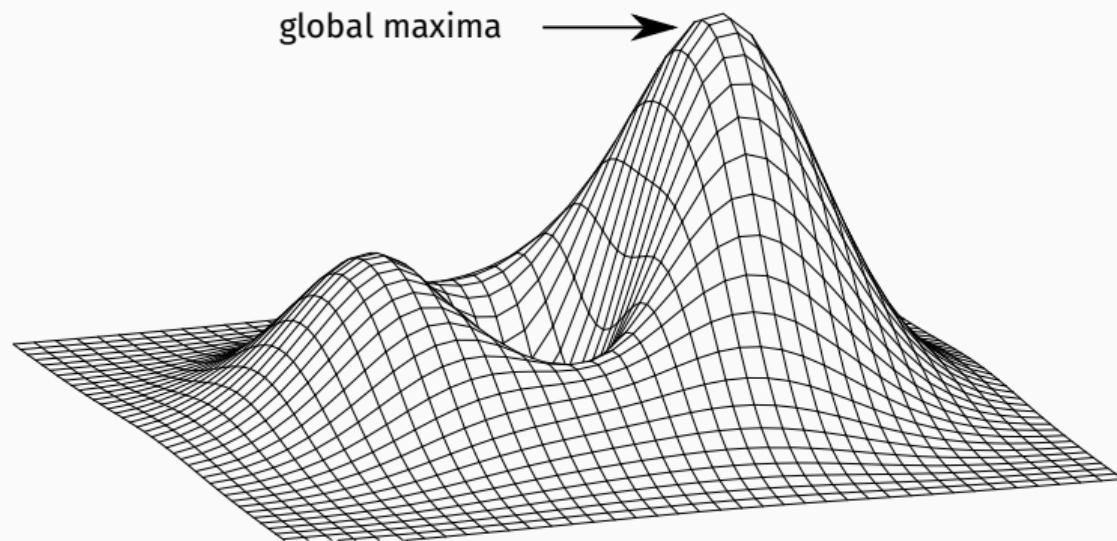
We **learn** a function that has the same I/O behavior:

$$h(x, y, z) := x + y + z$$

How to synthesize programs?

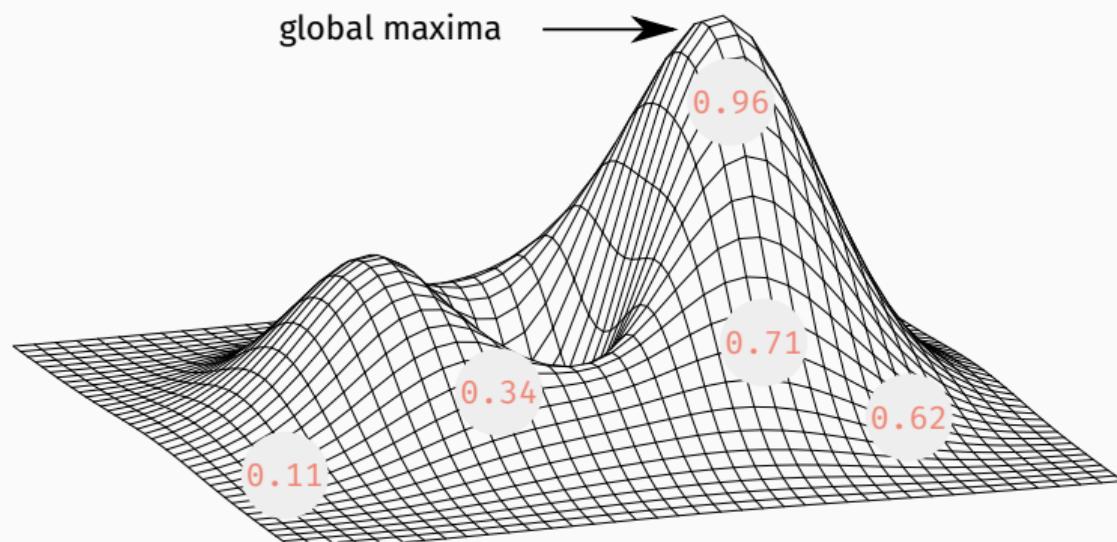
Stochastic Program Synthesis

- probabilistic optimization problem



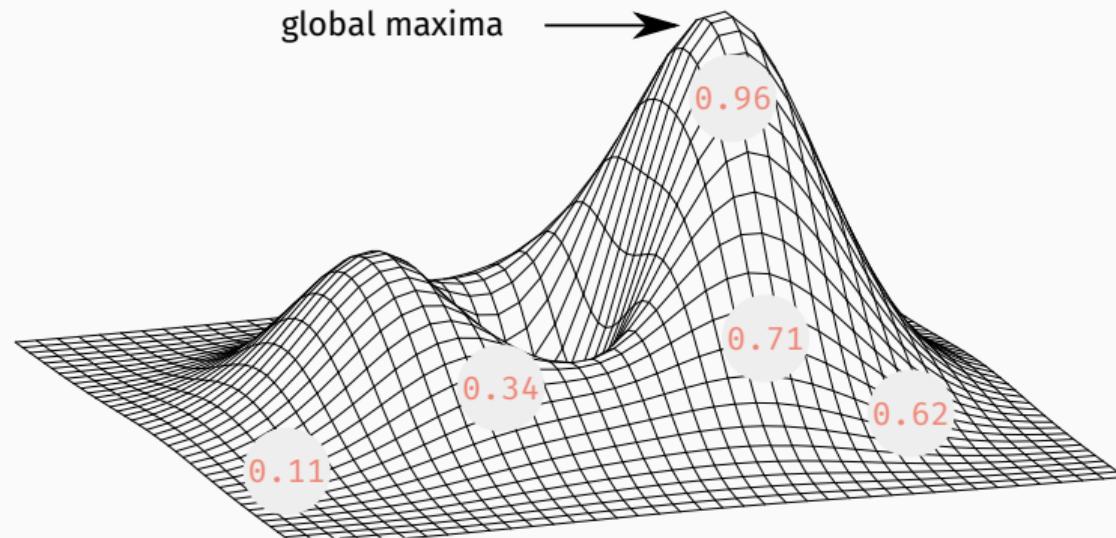
Stochastic Program Synthesis

- probabilistic optimization problem



Stochastic Program Synthesis

- probabilistic optimization problem
- based on Monte Carlo Tree Search (MCTS)



Let's synthesize: $a + b \bmod 8$

Program Generation

$$U \rightarrow U + U \mid U * U \mid a \mid b$$

Program Generation

$$U \rightarrow U + U \mid U * U \mid a \mid b$$

- non-terminal symbol: U

Program Generation

$$U \rightarrow U + U \mid U * U \mid a \mid b$$

- non-terminal symbol: U
- input variables: $\{a, b\}$

Program Generation

$$U \rightarrow U + U \mid U * U \mid a \mid b$$

- non-terminal symbol: U
- input variables: $\{a, b\}$
- candidate programs: $a, b, a * b, a + b, \dots$

Program Generation

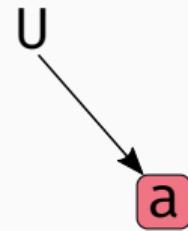
$$U \rightarrow U + U \mid U * U \mid a \mid b$$

- non-terminal symbol: U
- input variables: $\{a, b\}$
- candidate programs: $a, b, a * b, a + b, \dots$
- intermediate programs: $U + U, U * U, U + b, \dots$

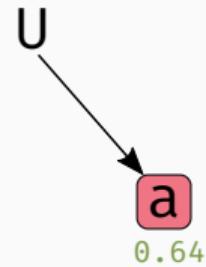
Monte Carlo Tree Search

U

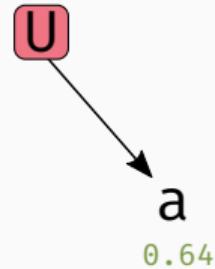
Monte Carlo Tree Search



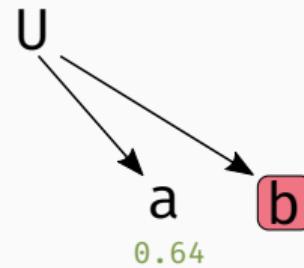
Monte Carlo Tree Search



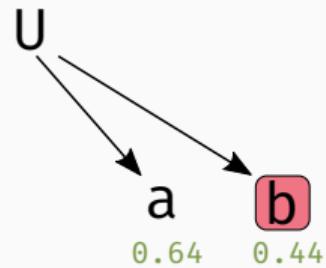
Monte Carlo Tree Search



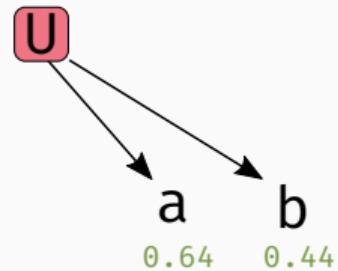
Monte Carlo Tree Search



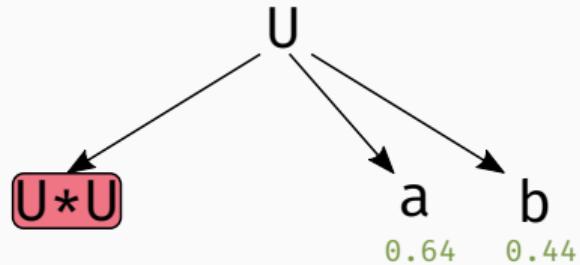
Monte Carlo Tree Search



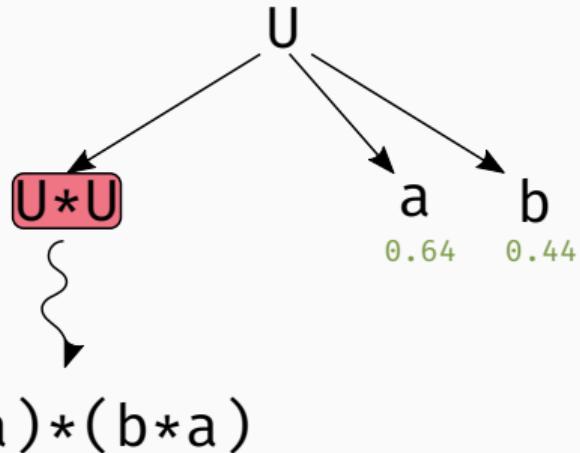
Monte Carlo Tree Search



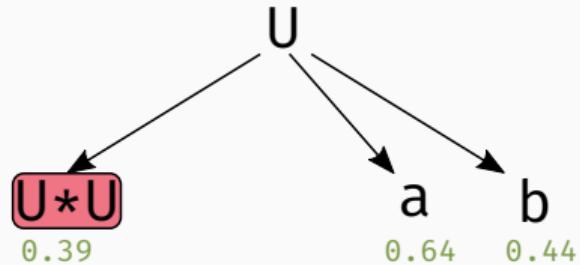
Monte Carlo Tree Search



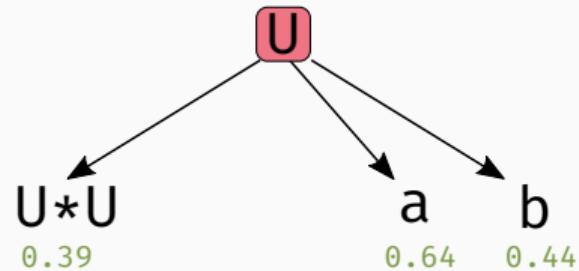
Monte Carlo Tree Search



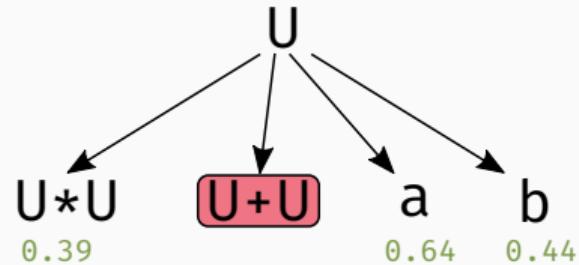
Monte Carlo Tree Search



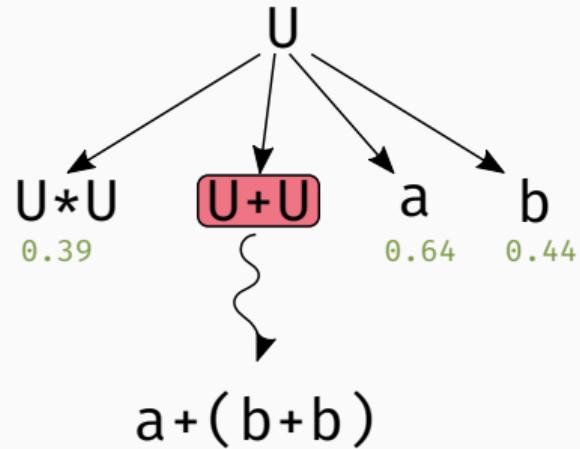
Monte Carlo Tree Search



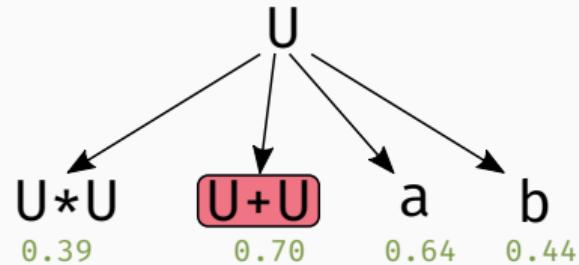
Monte Carlo Tree Search



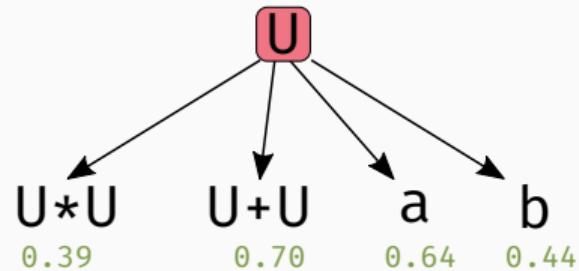
Monte Carlo Tree Search



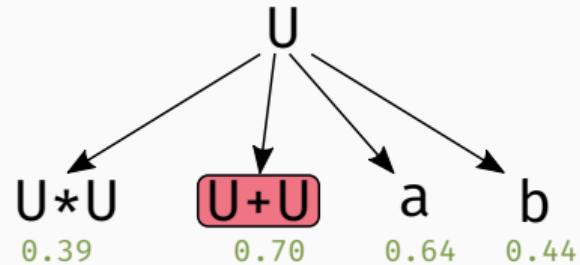
Monte Carlo Tree Search



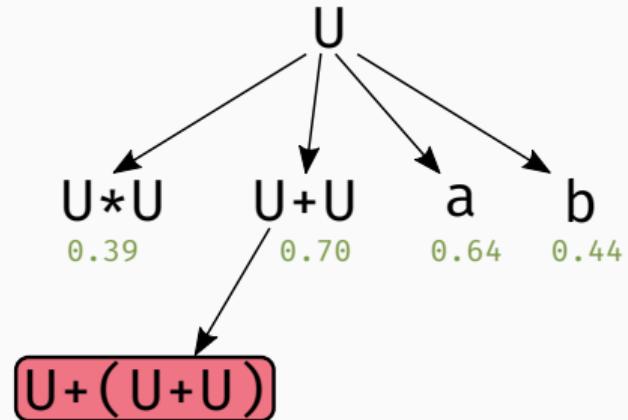
Monte Carlo Tree Search



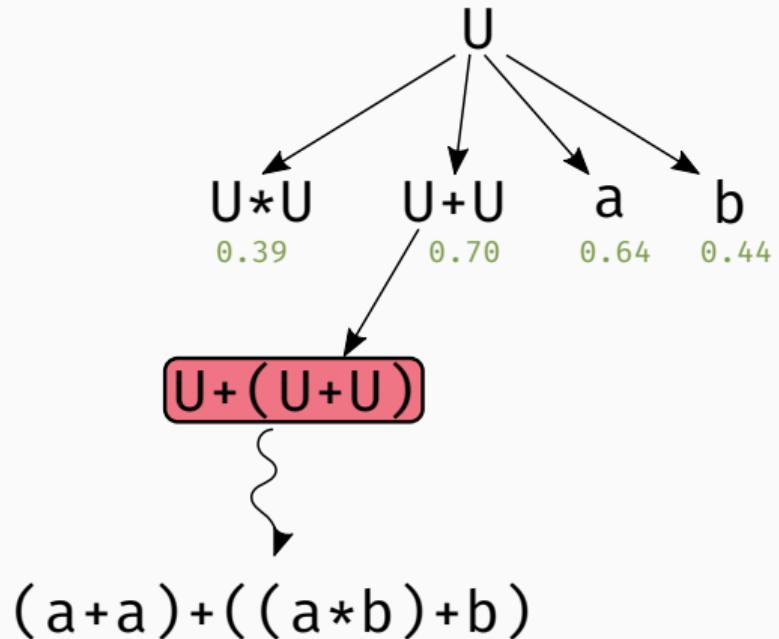
Monte Carlo Tree Search



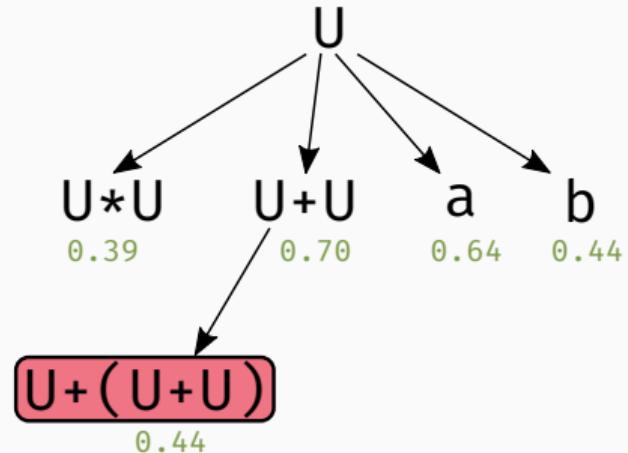
Monte Carlo Tree Search



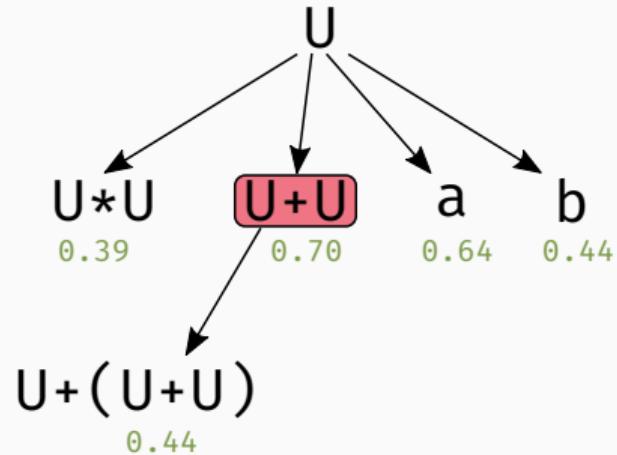
Monte Carlo Tree Search



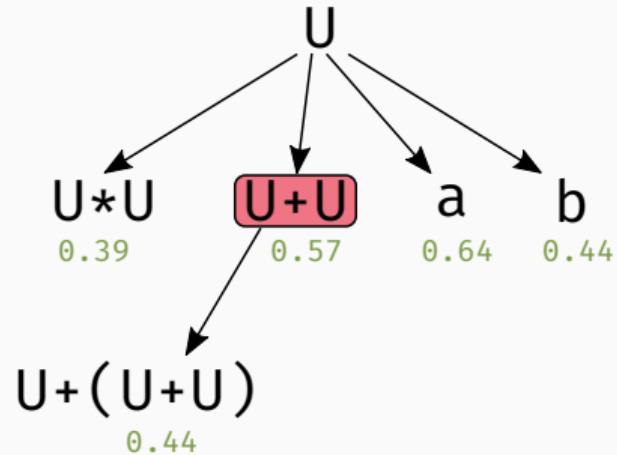
Monte Carlo Tree Search



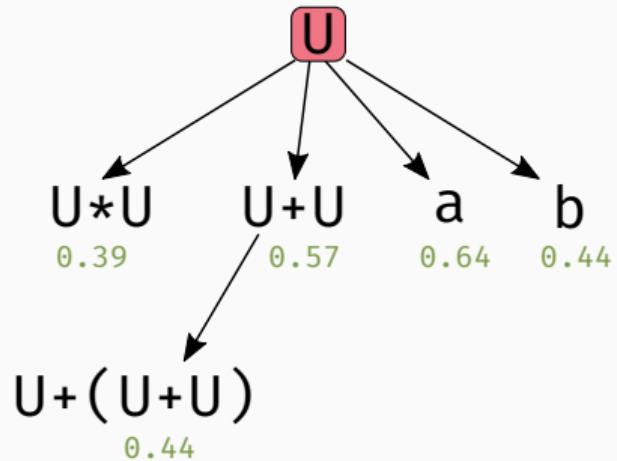
Monte Carlo Tree Search



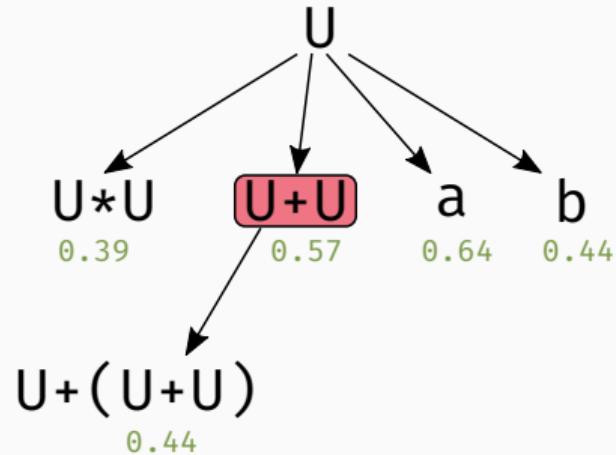
Monte Carlo Tree Search



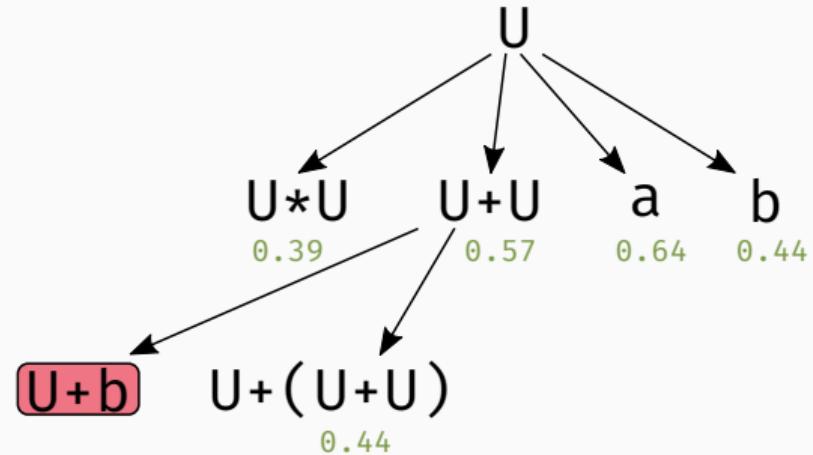
Monte Carlo Tree Search



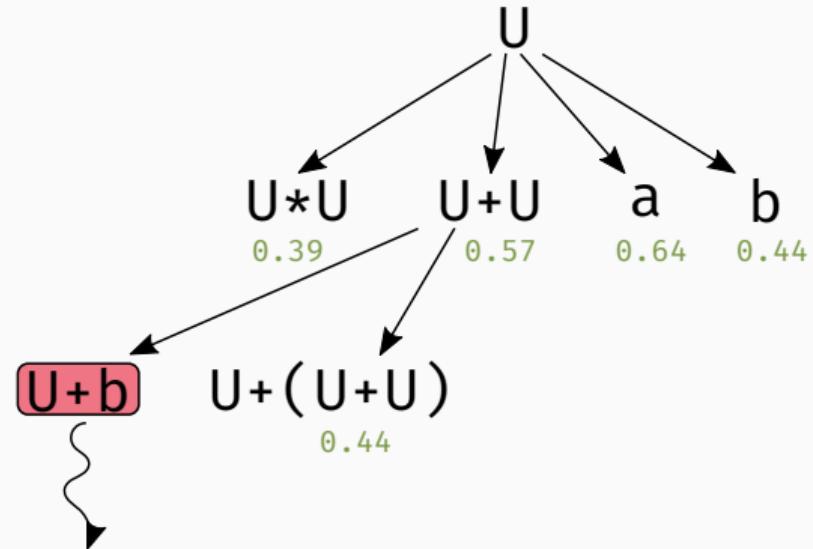
Monte Carlo Tree Search



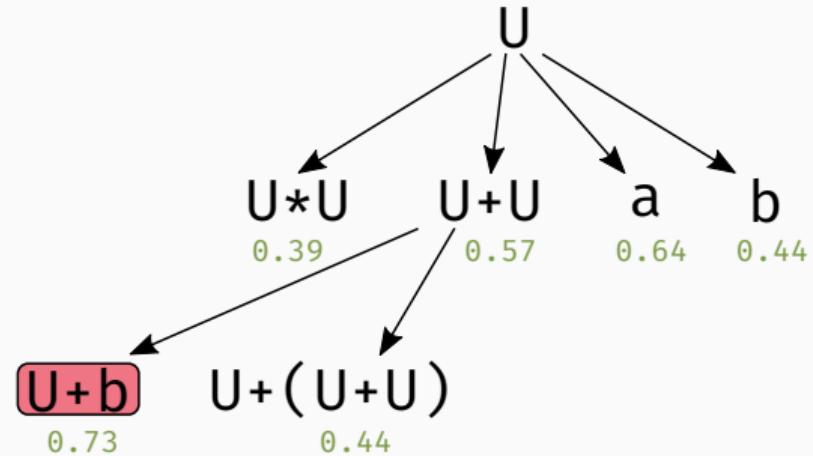
Monte Carlo Tree Search



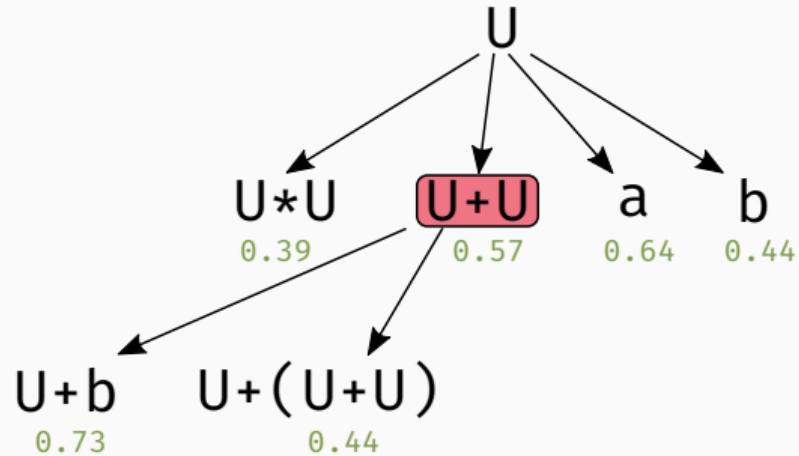
Monte Carlo Tree Search



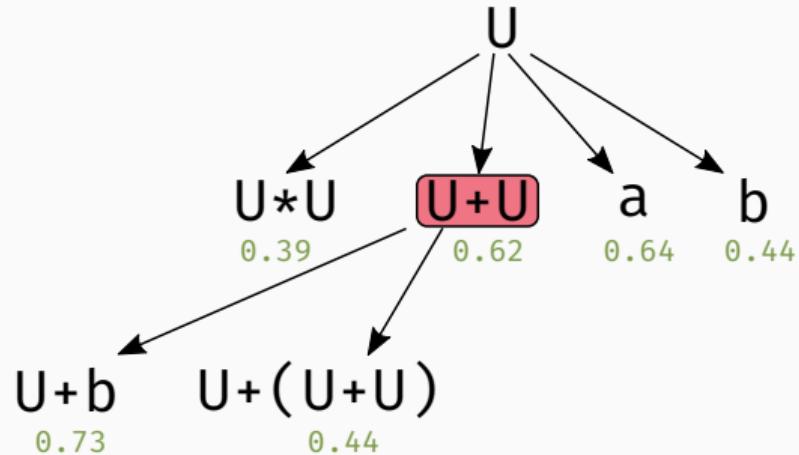
Monte Carlo Tree Search



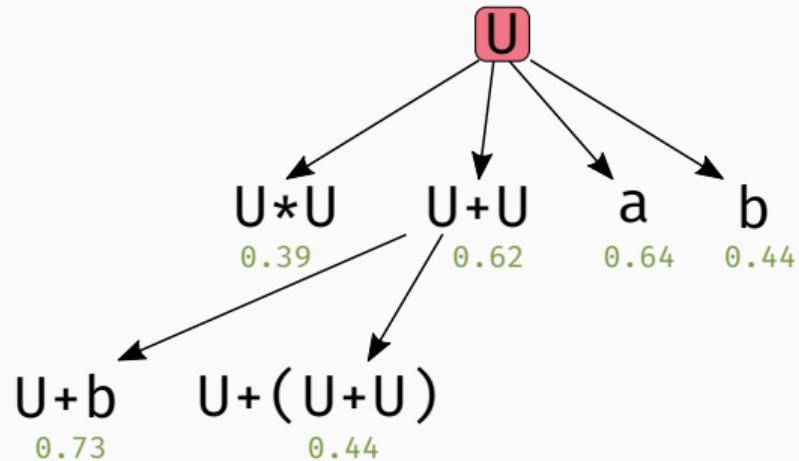
Monte Carlo Tree Search



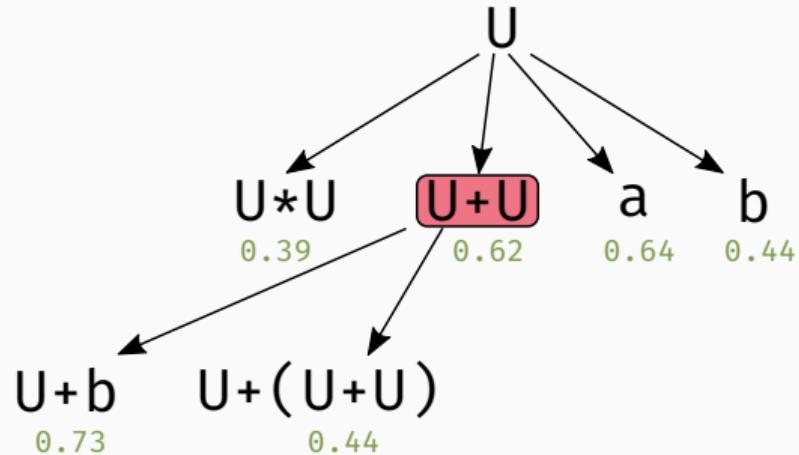
Monte Carlo Tree Search



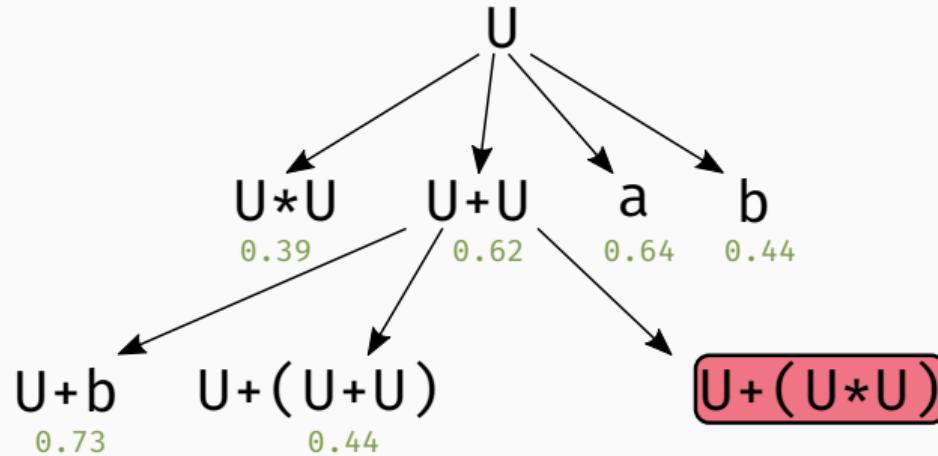
Monte Carlo Tree Search



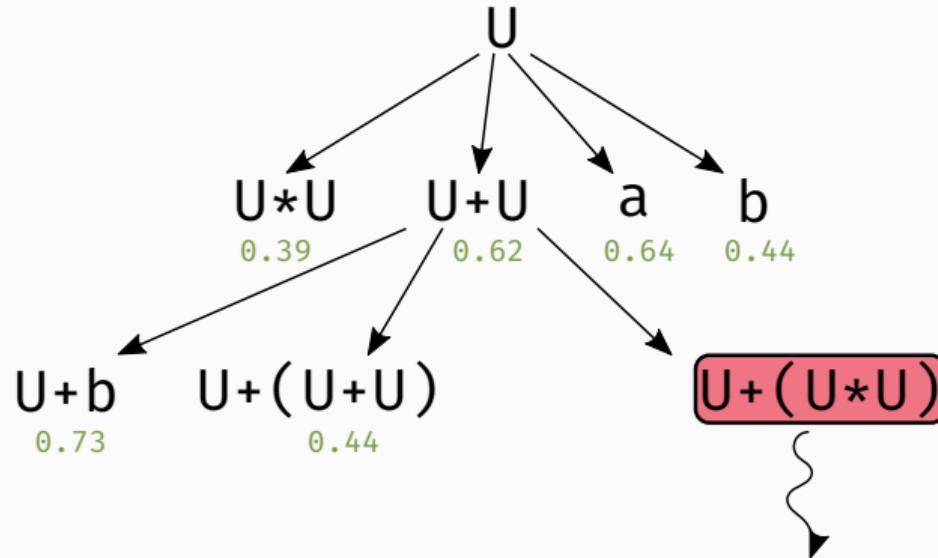
Monte Carlo Tree Search



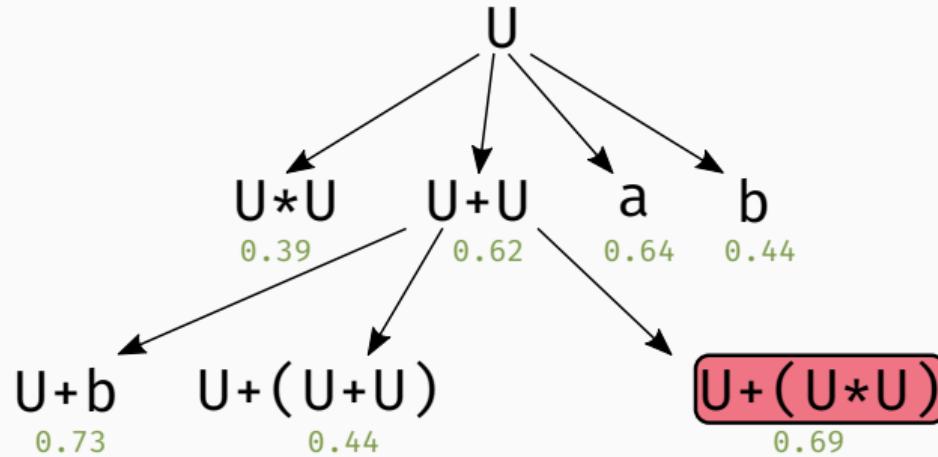
Monte Carlo Tree Search



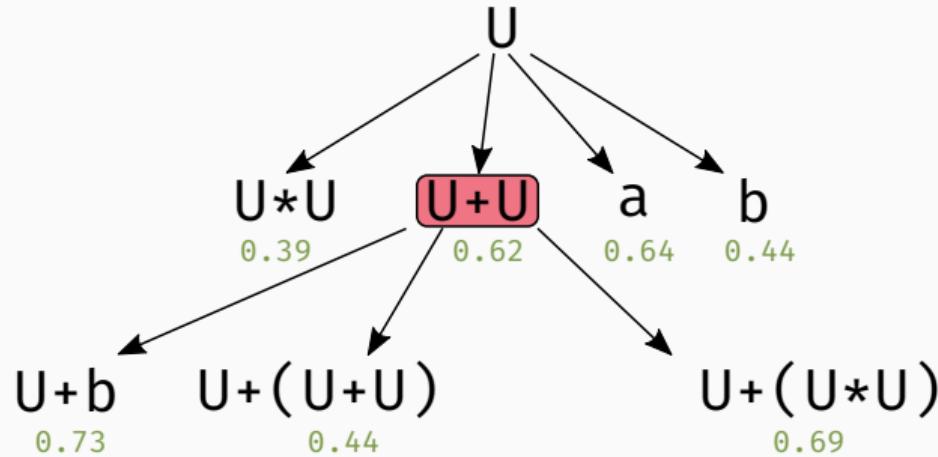
Monte Carlo Tree Search



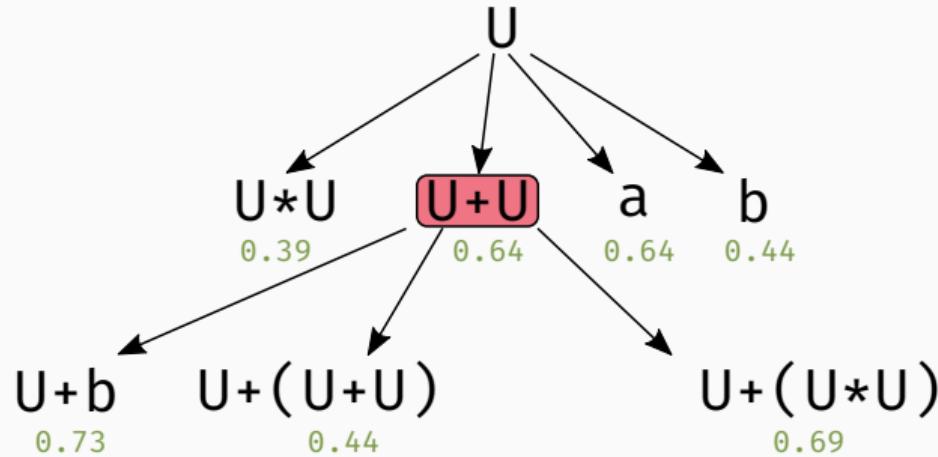
Monte Carlo Tree Search



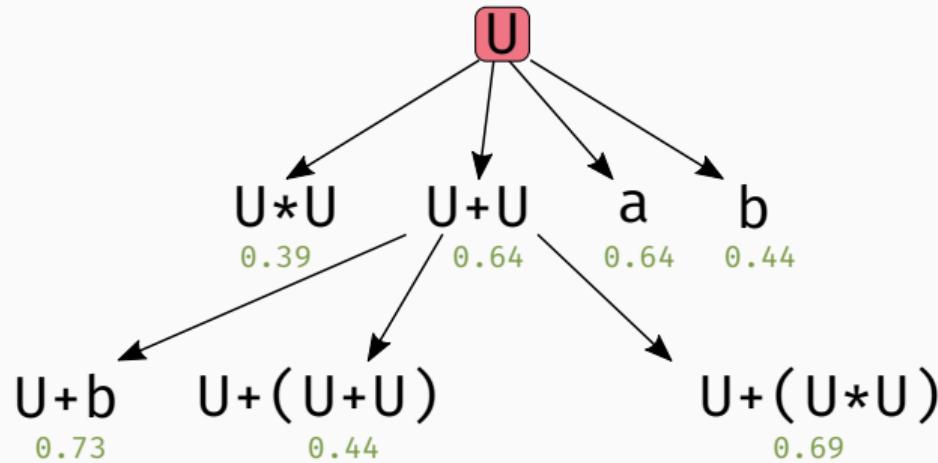
Monte Carlo Tree Search



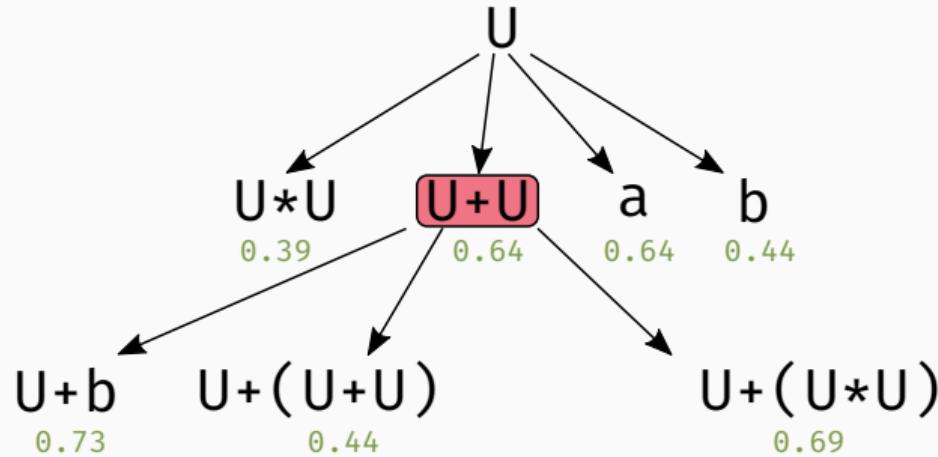
Monte Carlo Tree Search



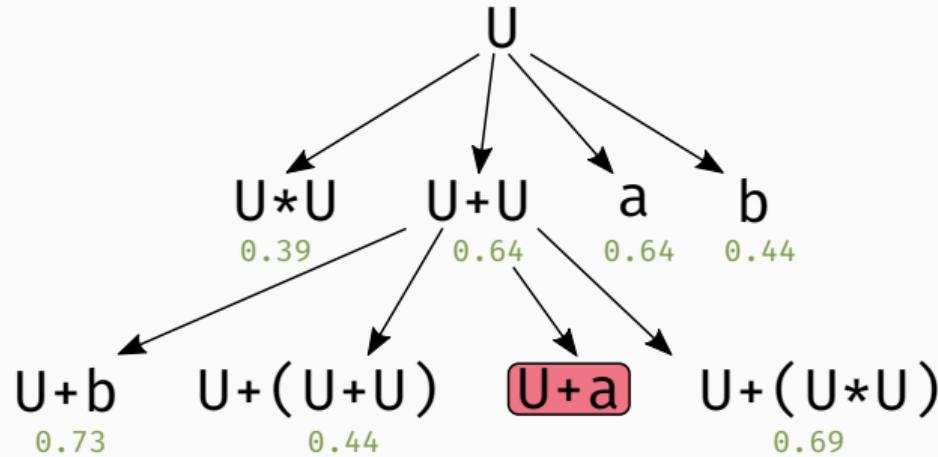
Monte Carlo Tree Search



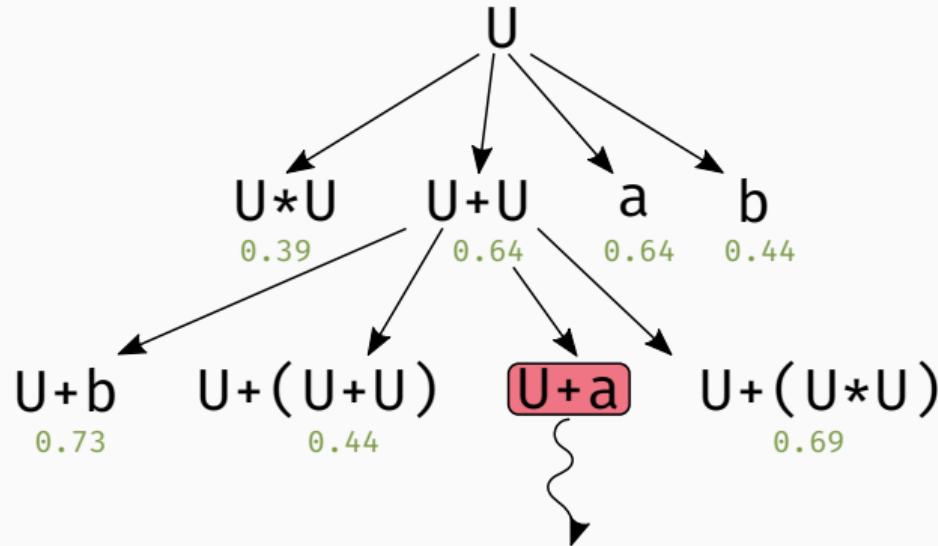
Monte Carlo Tree Search



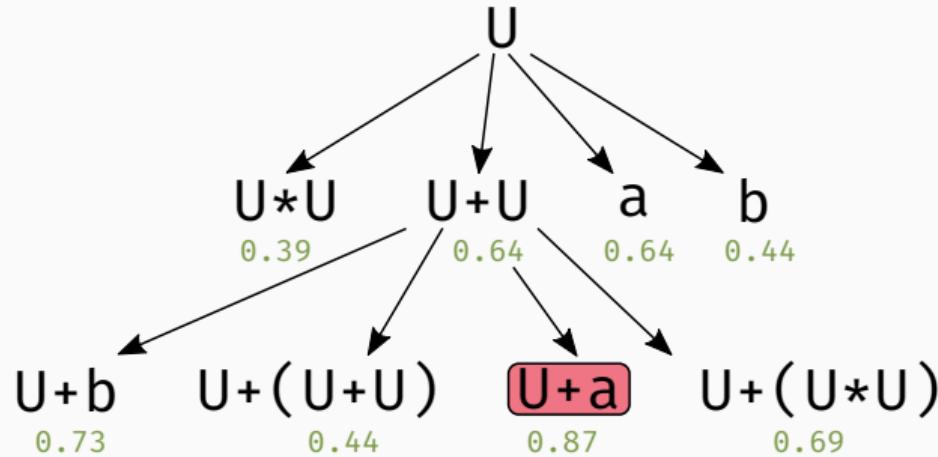
Monte Carlo Tree Search



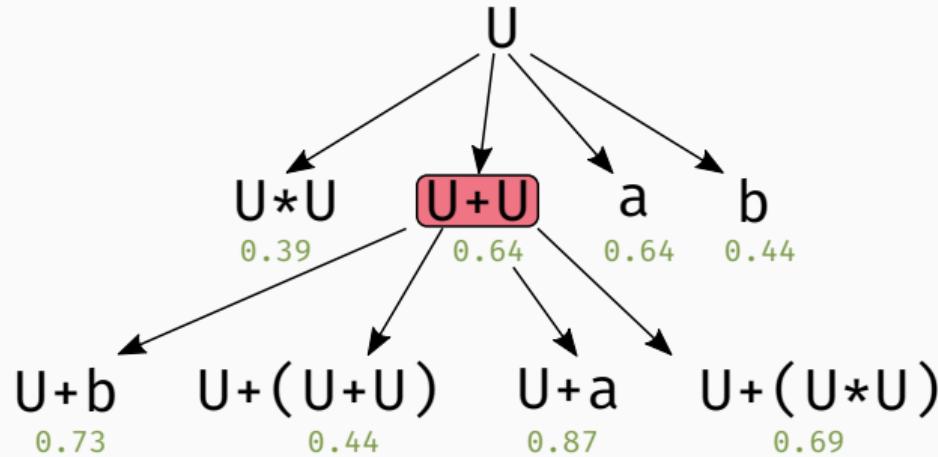
Monte Carlo Tree Search



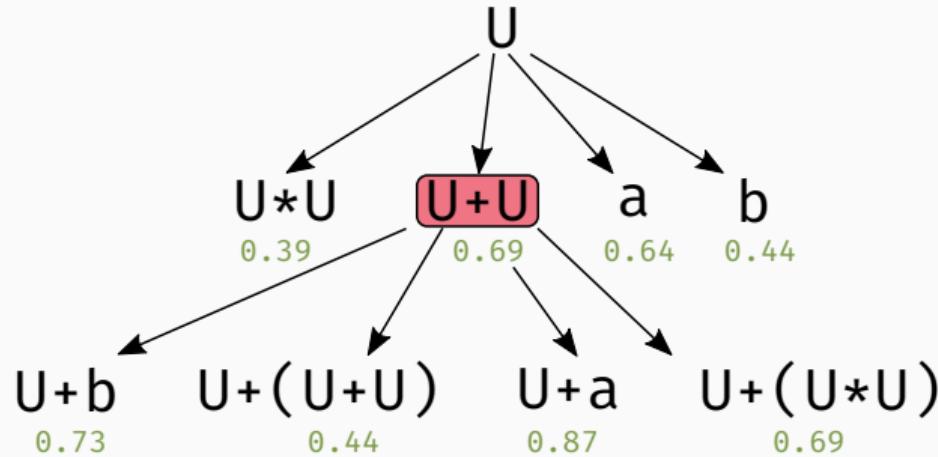
Monte Carlo Tree Search



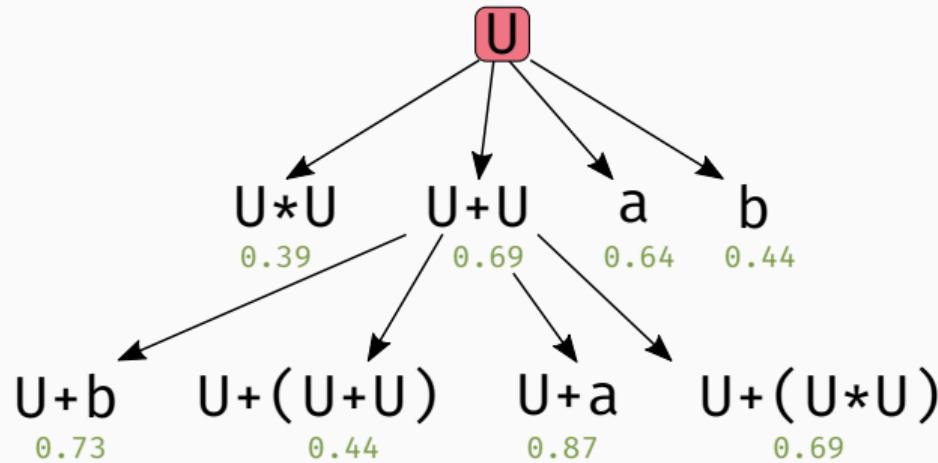
Monte Carlo Tree Search



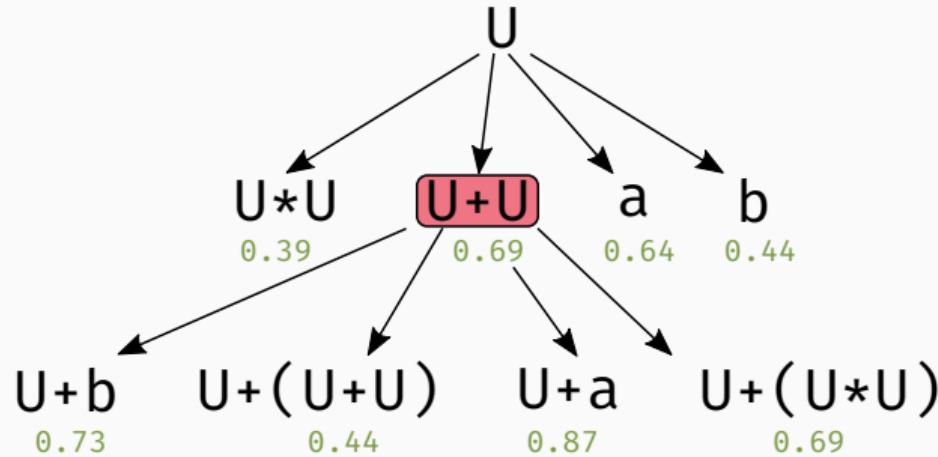
Monte Carlo Tree Search



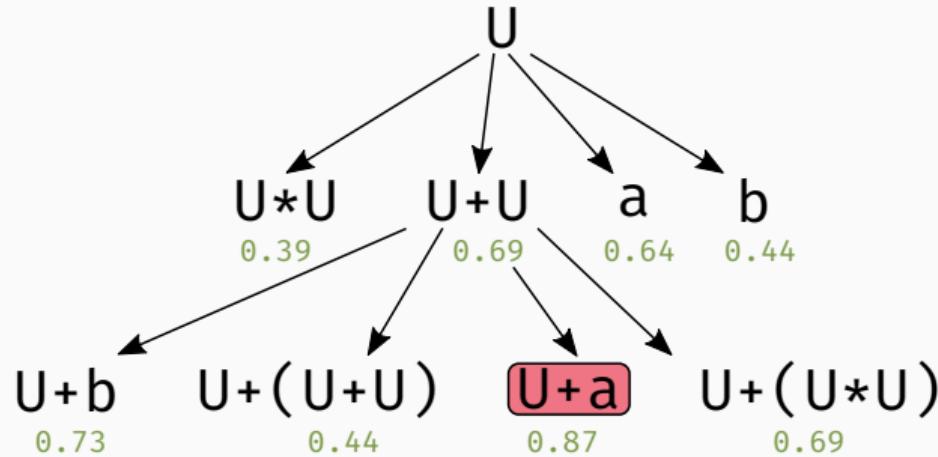
Monte Carlo Tree Search



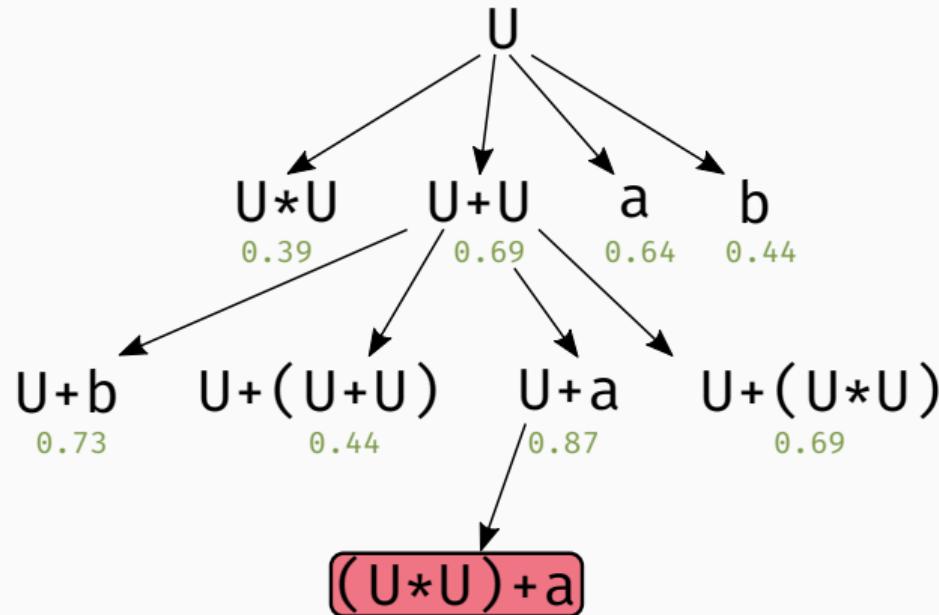
Monte Carlo Tree Search



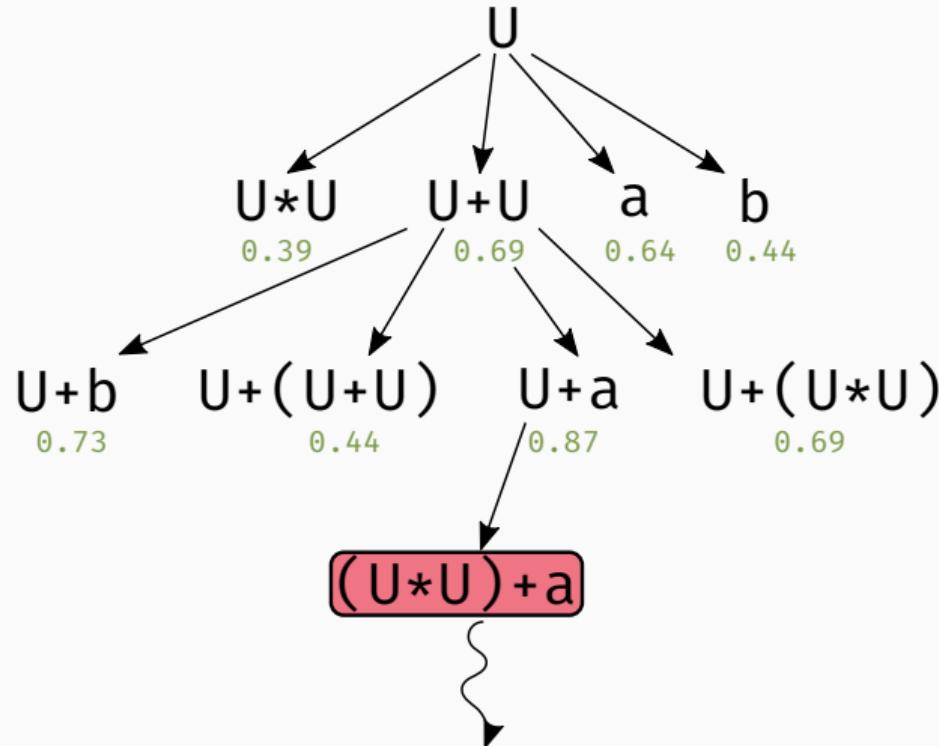
Monte Carlo Tree Search



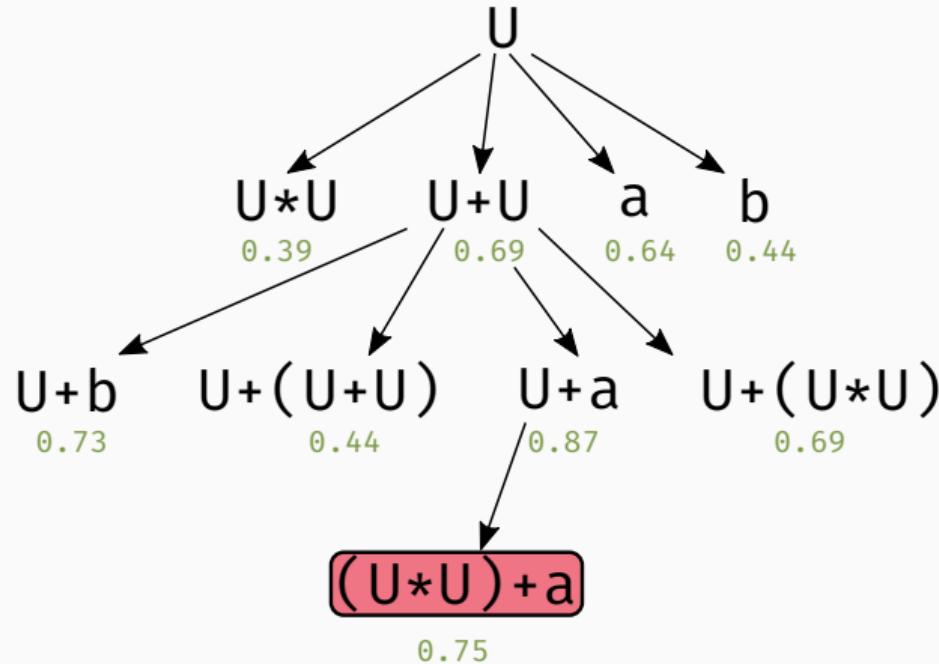
Monte Carlo Tree Search



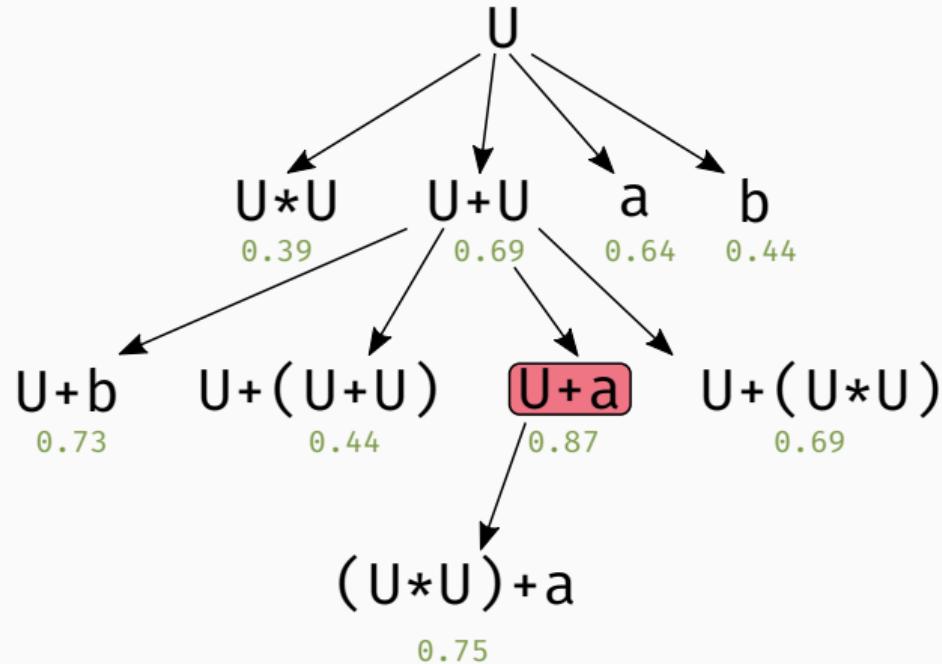
Monte Carlo Tree Search



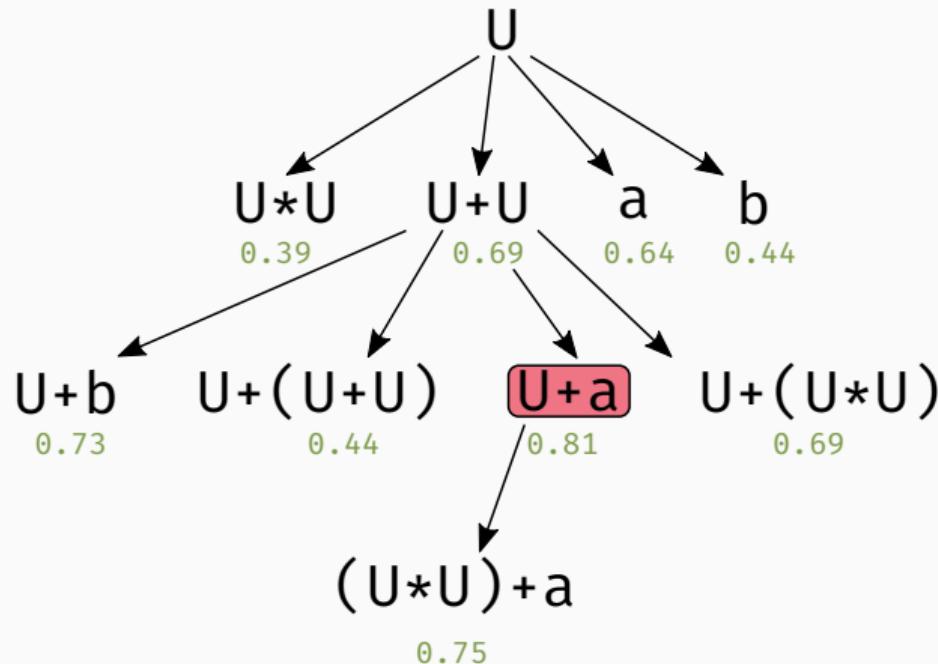
Monte Carlo Tree Search



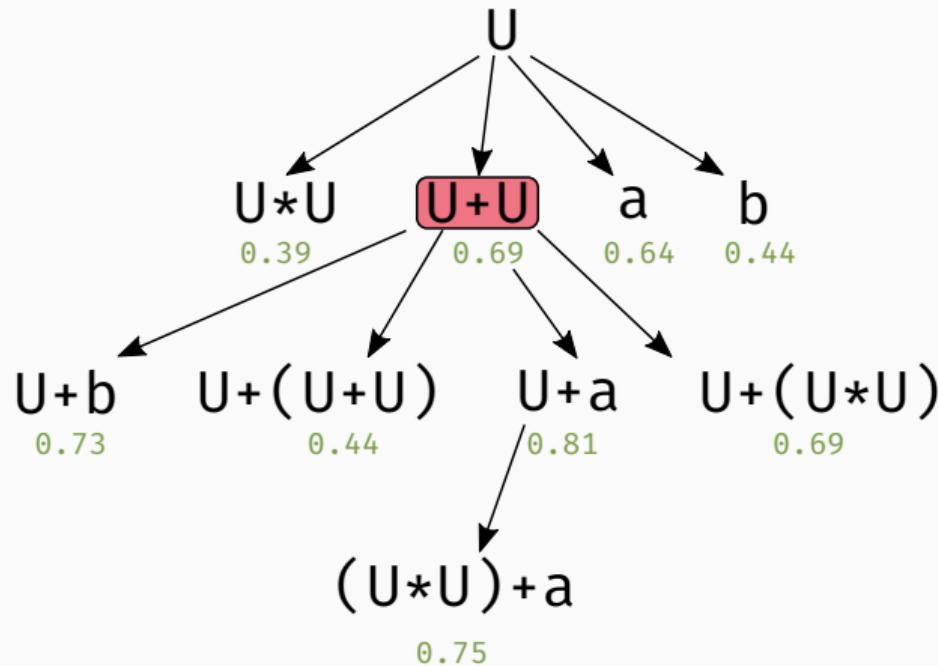
Monte Carlo Tree Search



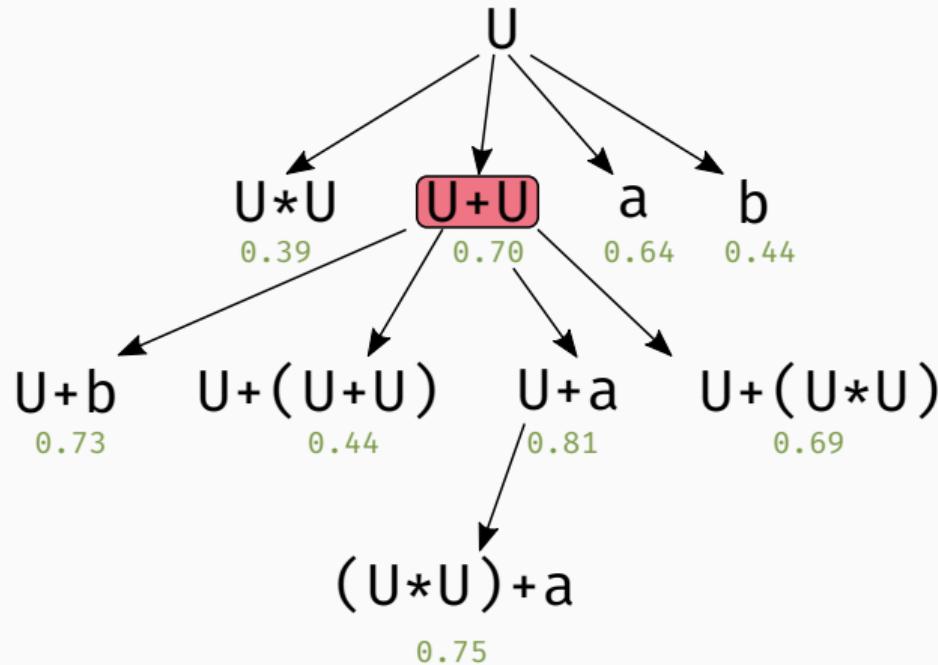
Monte Carlo Tree Search



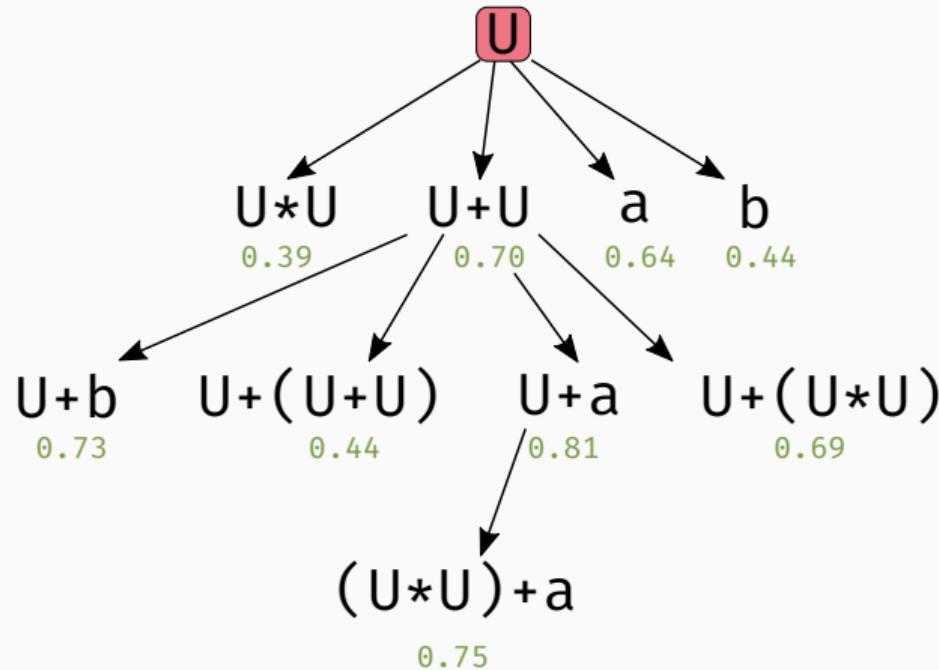
Monte Carlo Tree Search



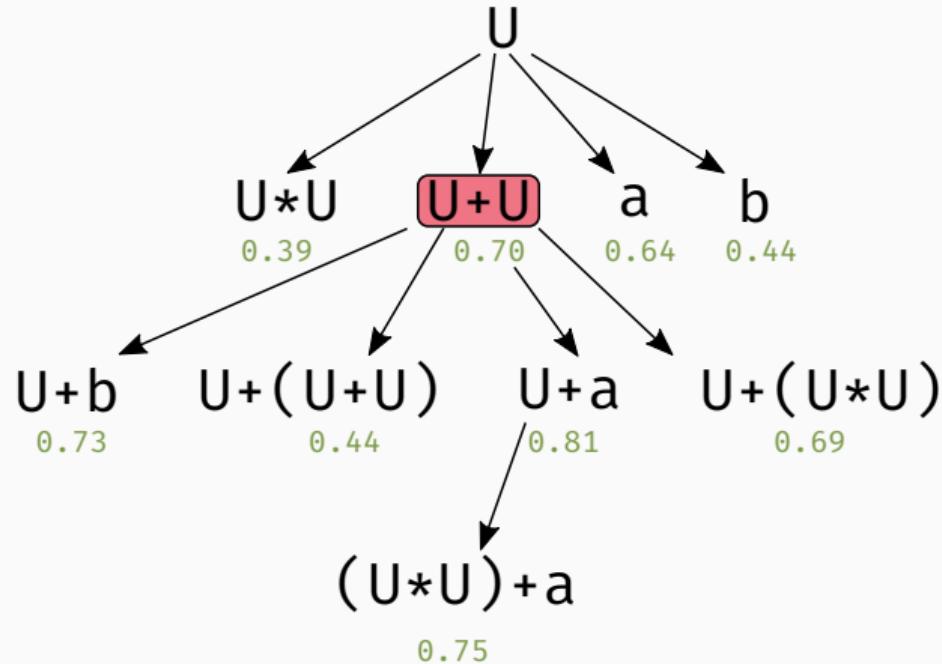
Monte Carlo Tree Search



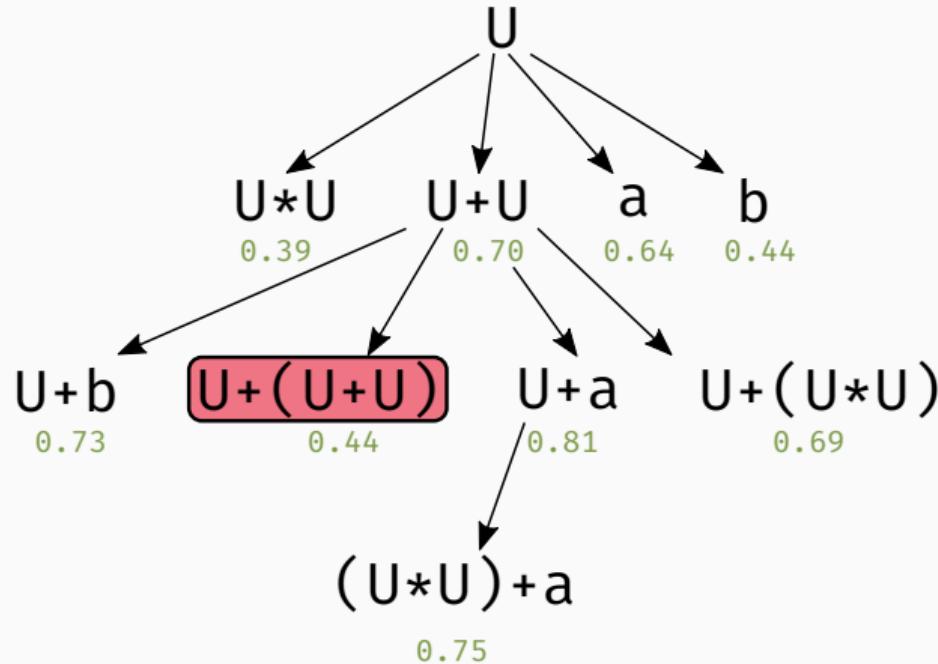
Monte Carlo Tree Search



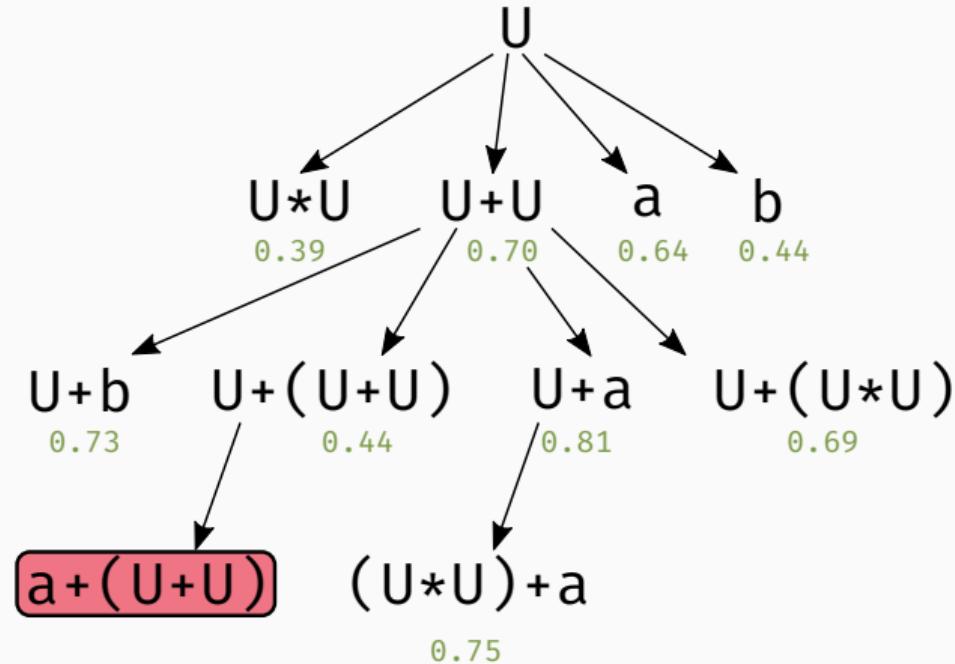
Monte Carlo Tree Search



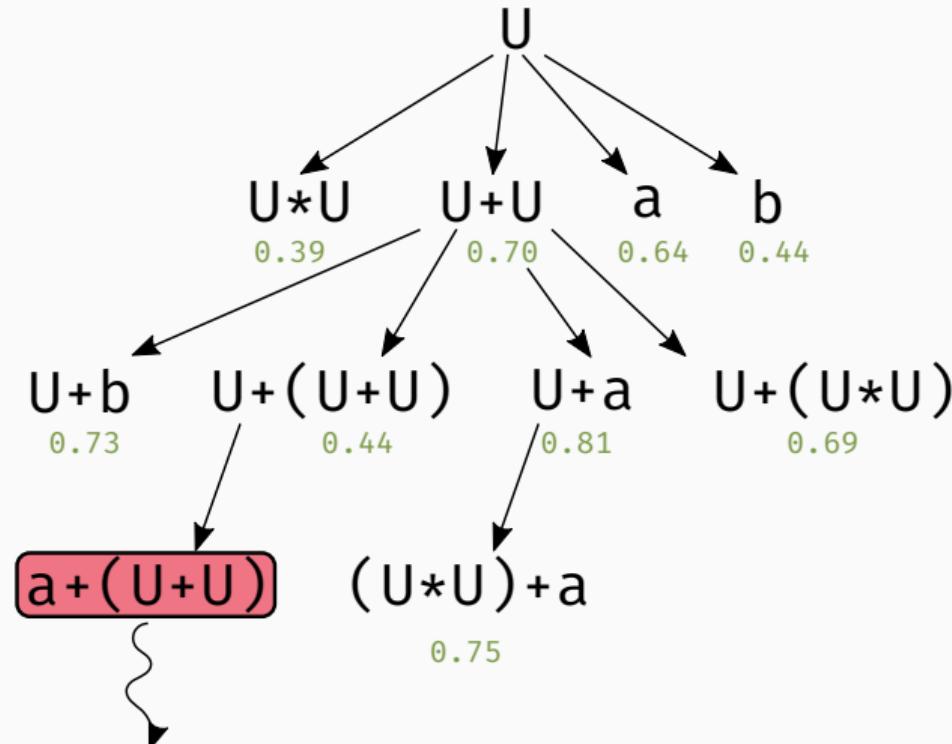
Monte Carlo Tree Search



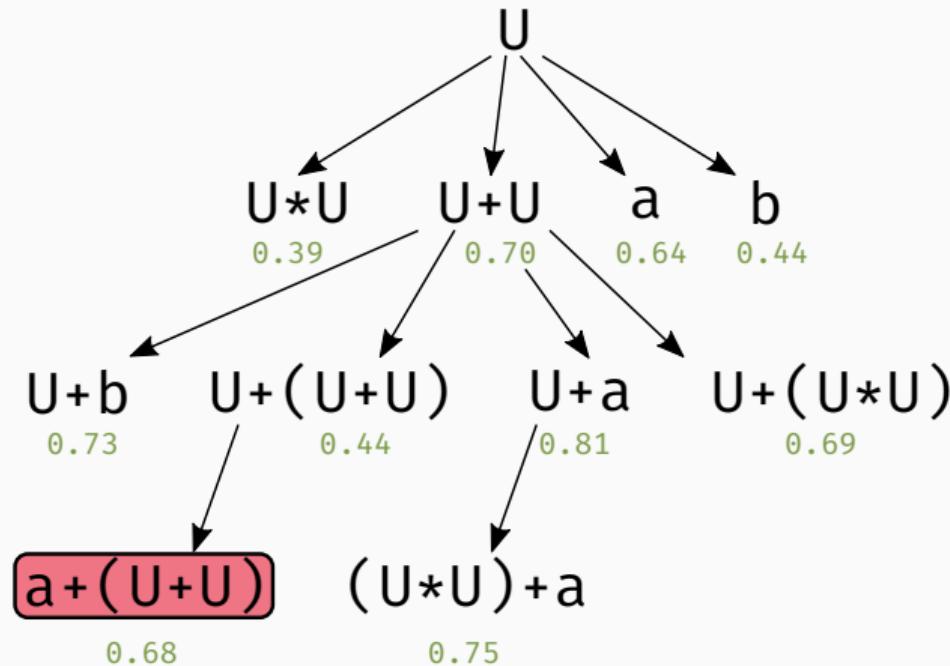
Monte Carlo Tree Search



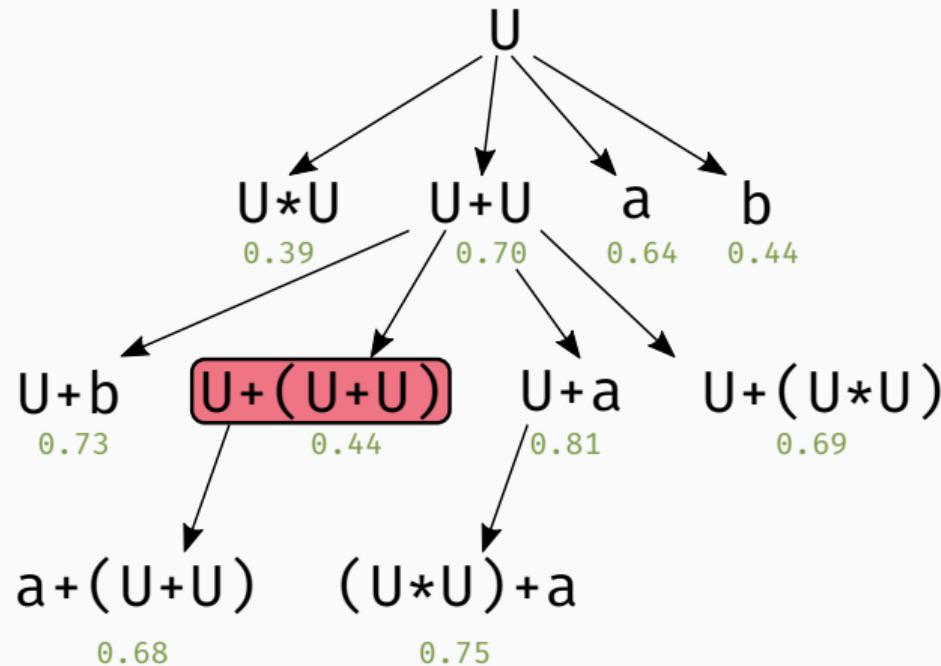
Monte Carlo Tree Search



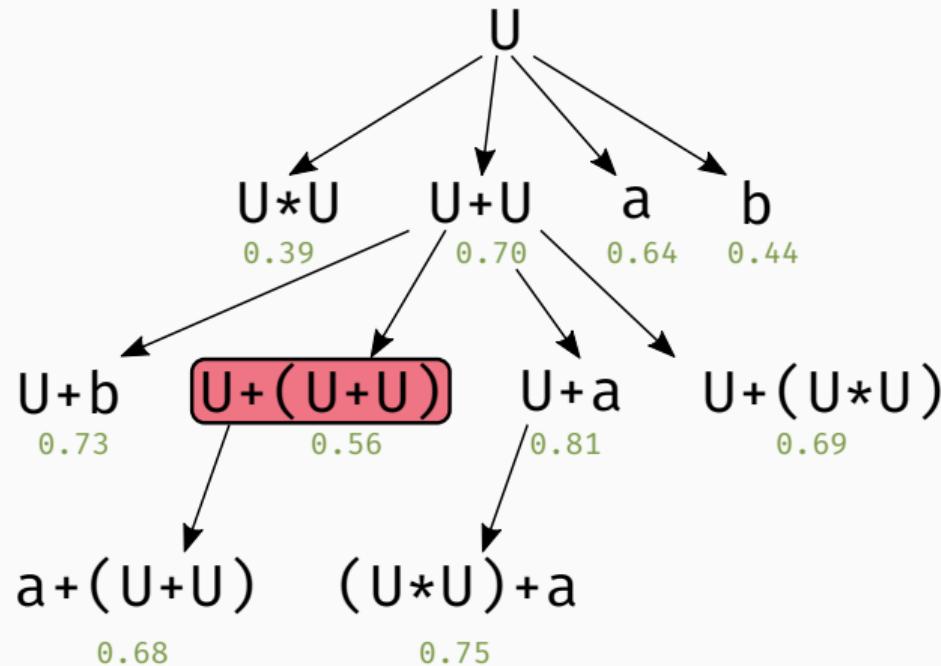
Monte Carlo Tree Search



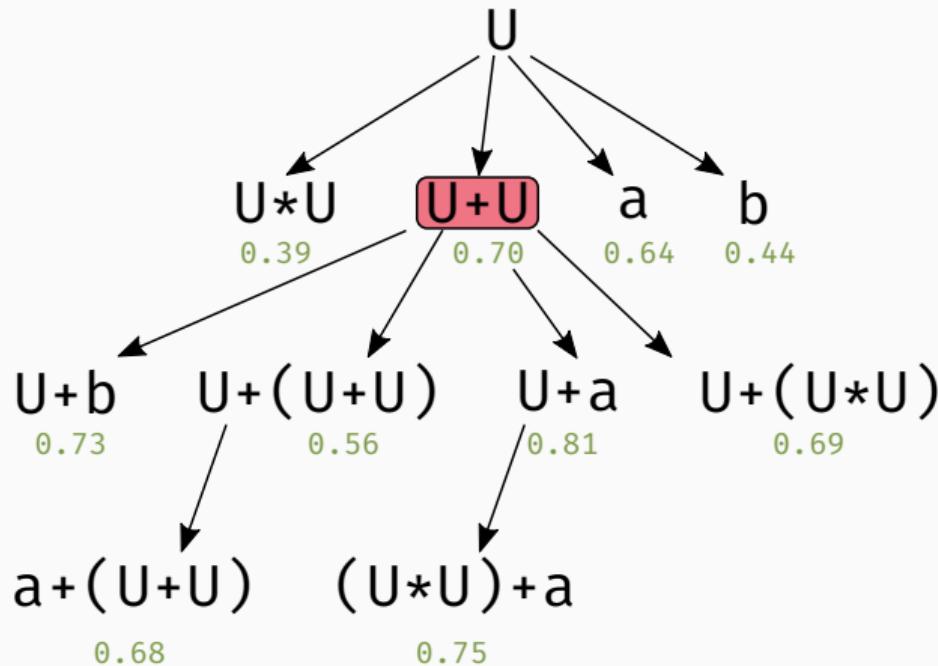
Monte Carlo Tree Search



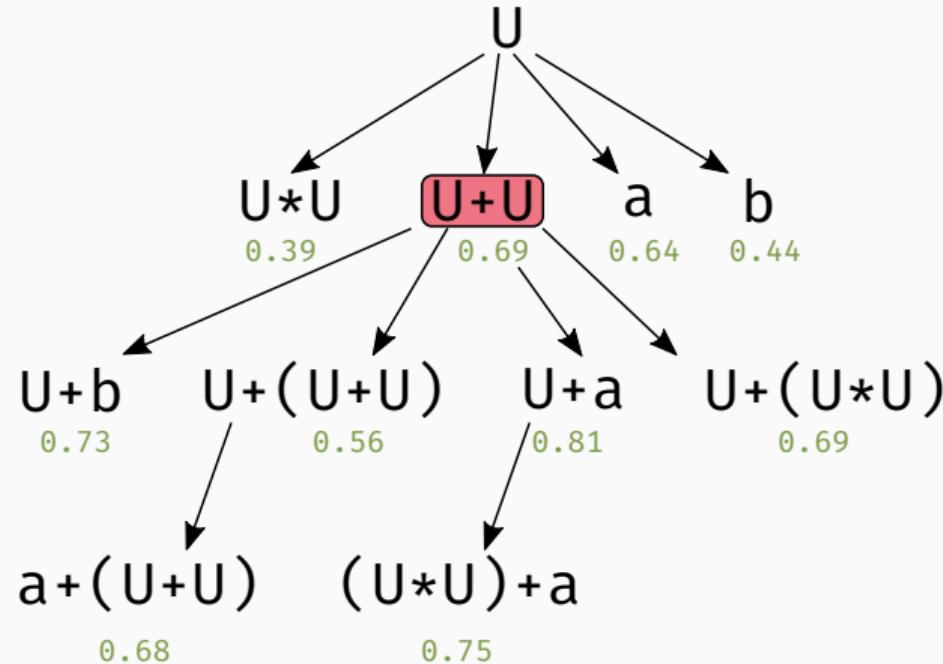
Monte Carlo Tree Search



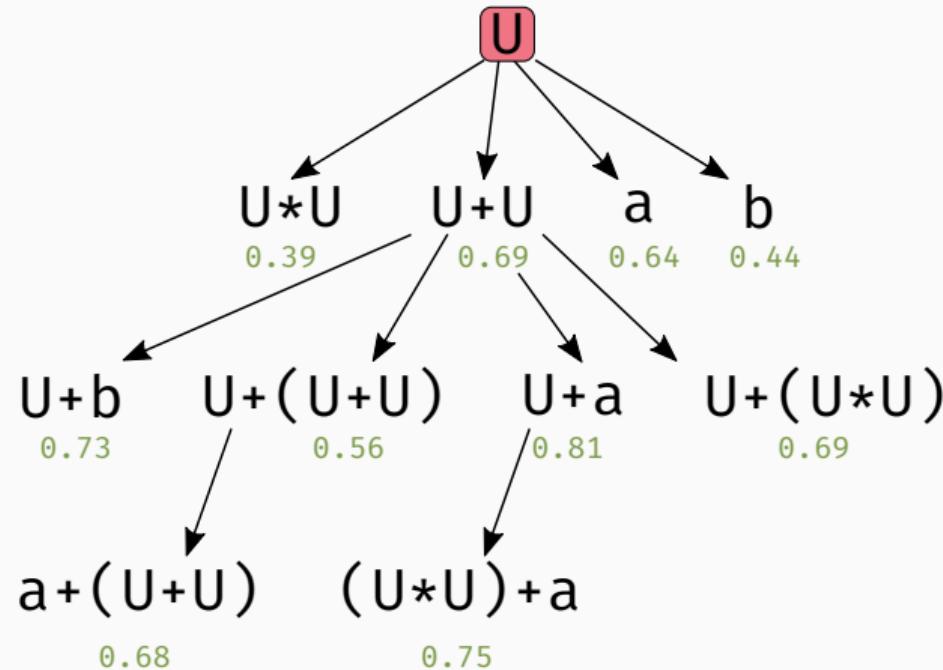
Monte Carlo Tree Search



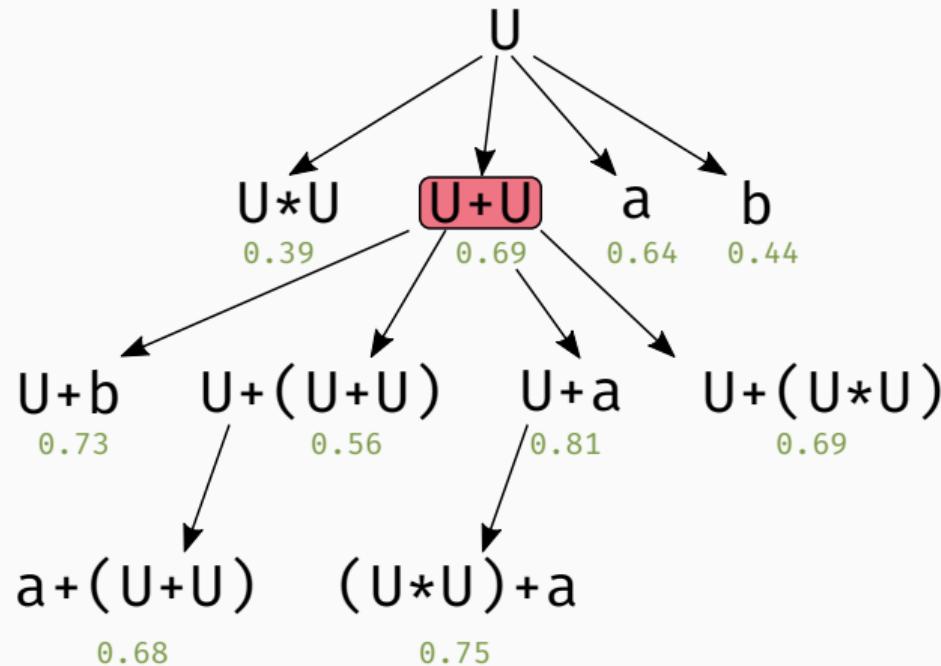
Monte Carlo Tree Search



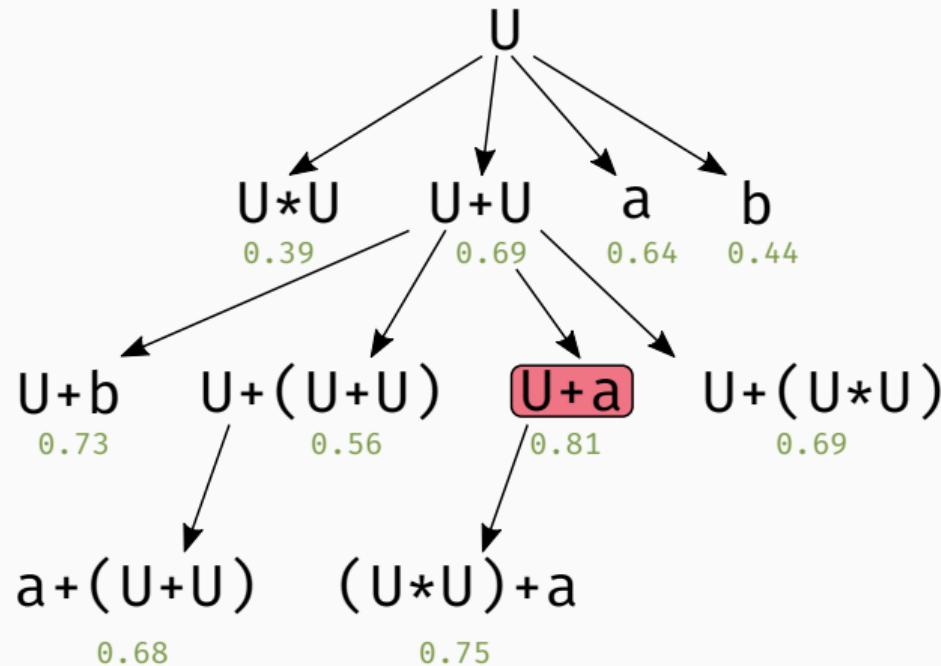
Monte Carlo Tree Search



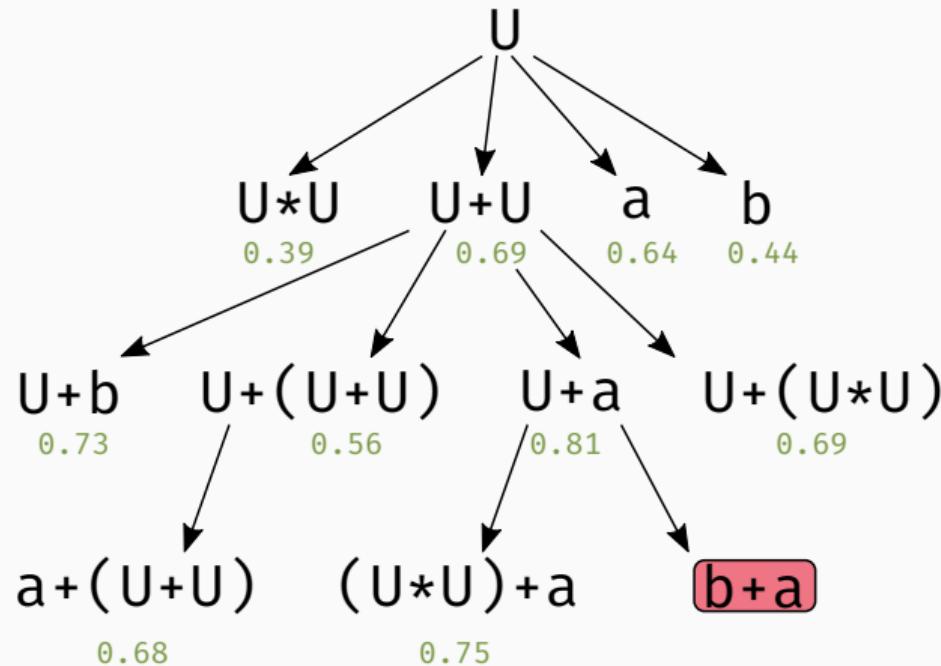
Monte Carlo Tree Search



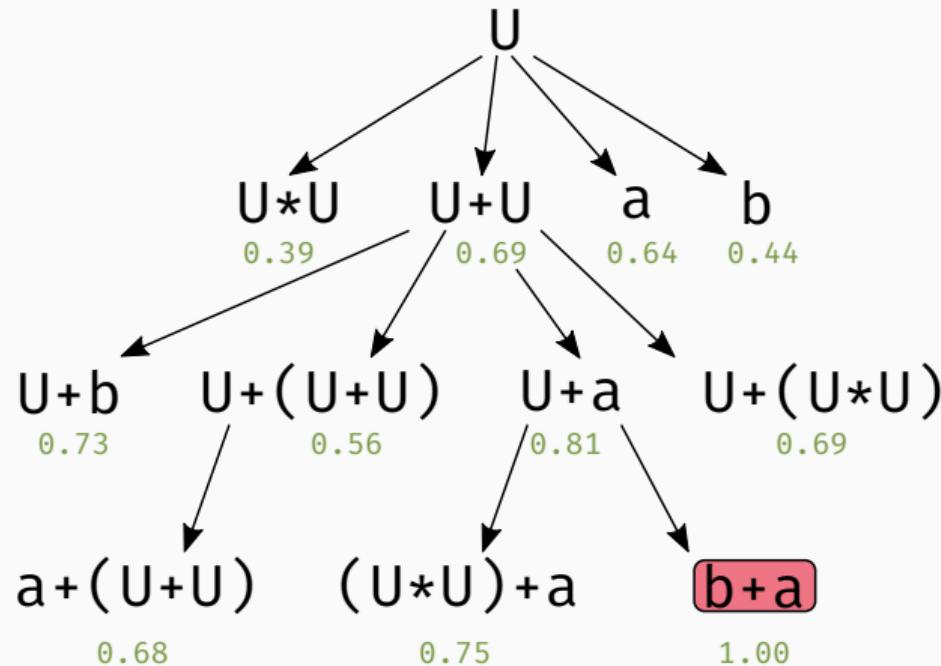
Monte Carlo Tree Search



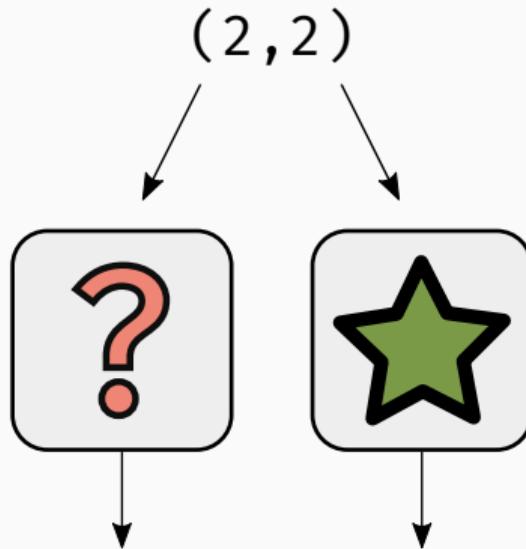
Monte Carlo Tree Search



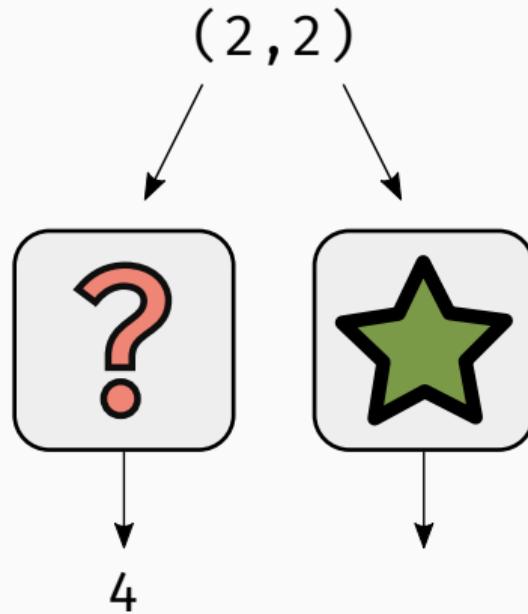
Monte Carlo Tree Search



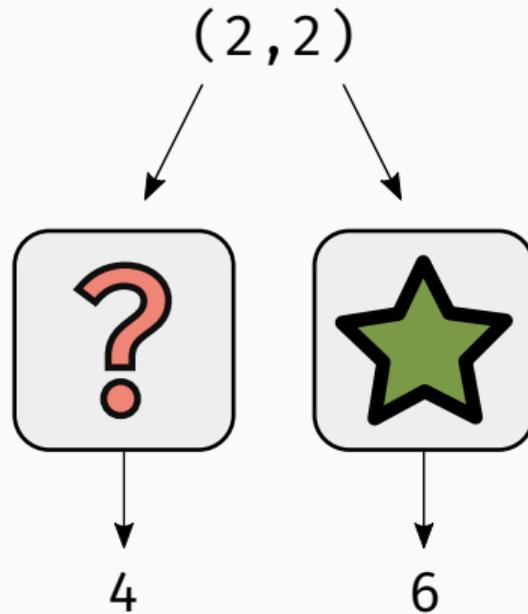
Score Calculation



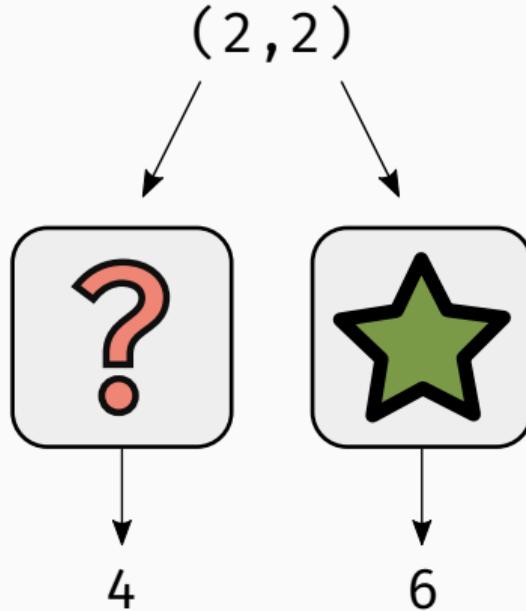
Score Calculation



Score Calculation

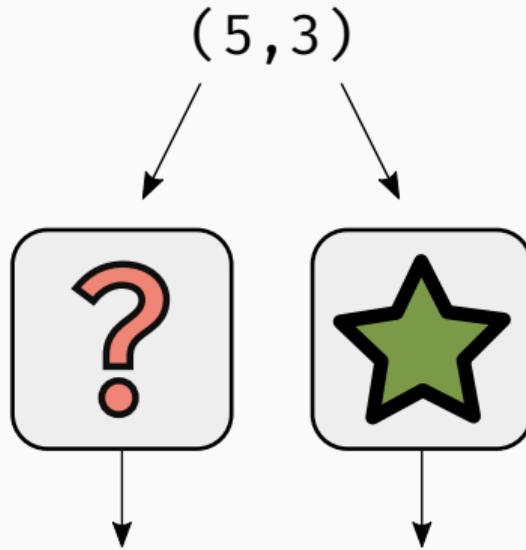


Score Calculation



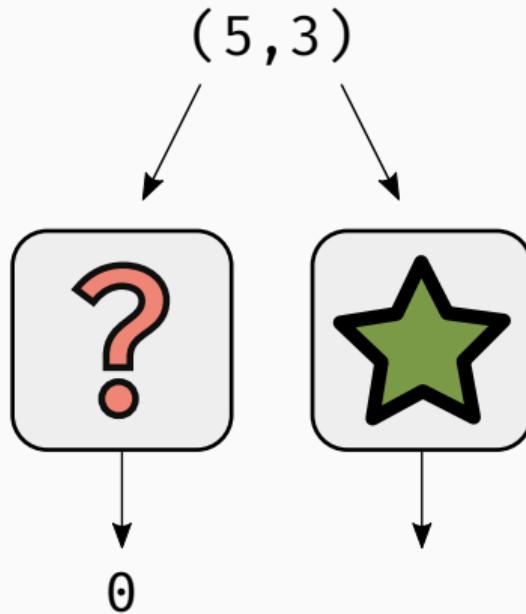
$\text{similarity}(4, 6) = 0.78$

Score Calculation



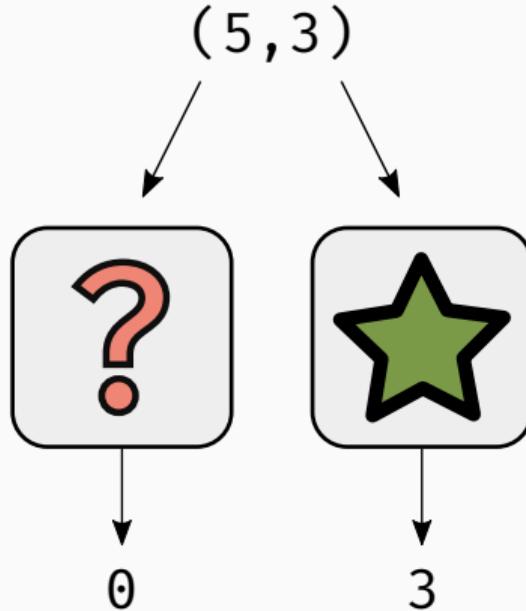
$\text{similarity}(4, 6) = 0.78$

Score Calculation



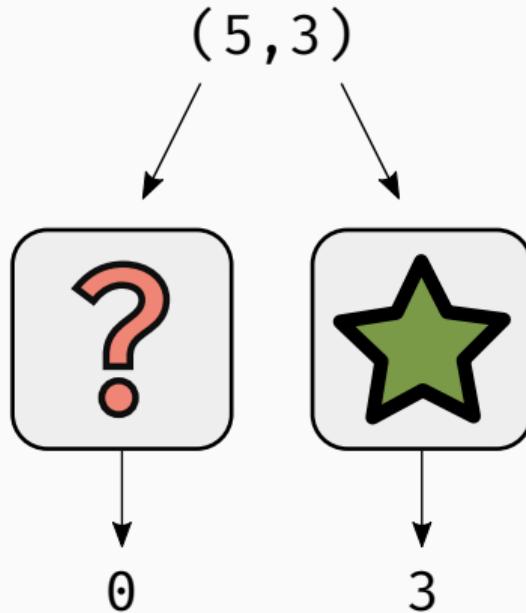
$$\text{similarity}(4, 6) = 0.78$$

Score Calculation



$\text{similarity}(4, 6) = 0.78$

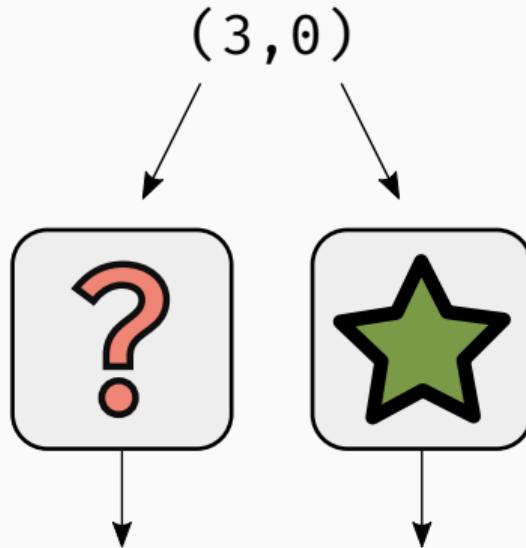
Score Calculation



$$\text{similarity}(4, 6) = 0.78$$

$$\text{similarity}(0, 3) = 0.33$$

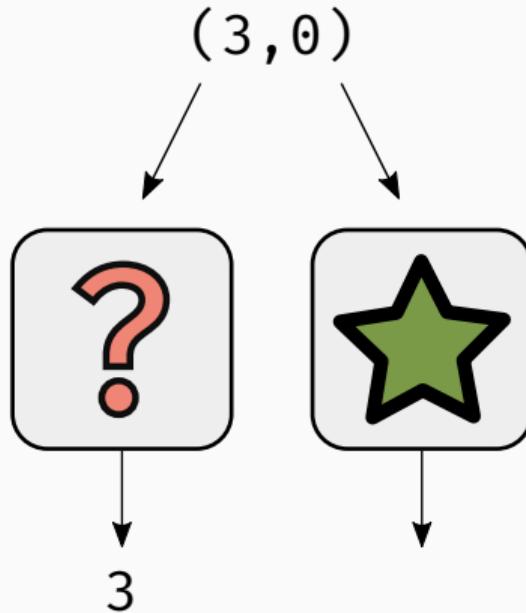
Score Calculation



$$\text{similarity}(4, 6) = 0.78$$

$$\text{similarity}(0, 3) = 0.33$$

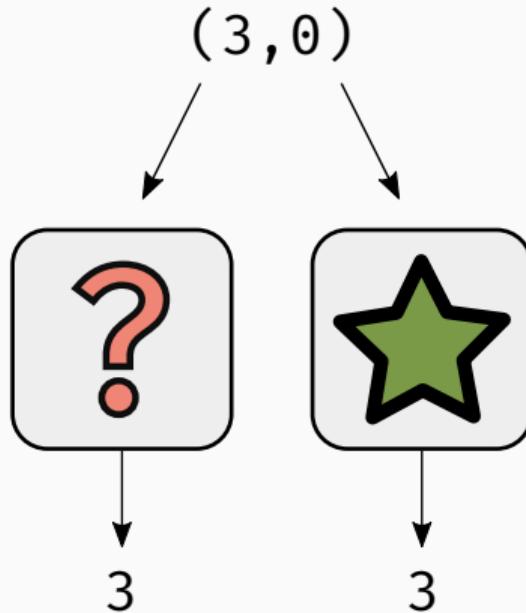
Score Calculation



$$\text{similarity}(4, 6) = 0.78$$

$$\text{similarity}(0, 3) = 0.33$$

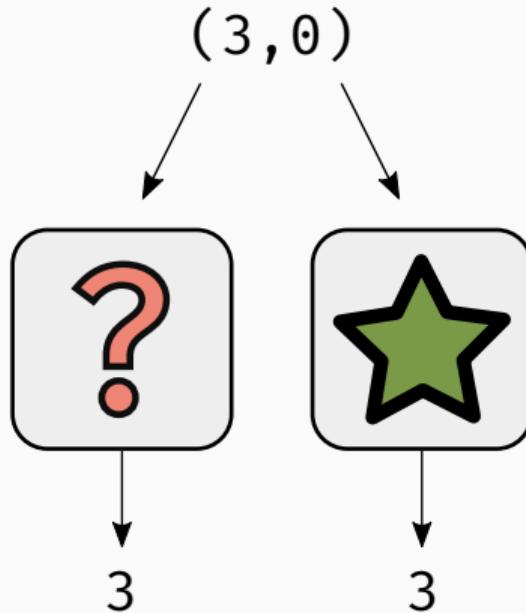
Score Calculation



$$\text{similarity}(4, 6) = 0.78$$

$$\text{similarity}(0, 3) = 0.33$$

Score Calculation

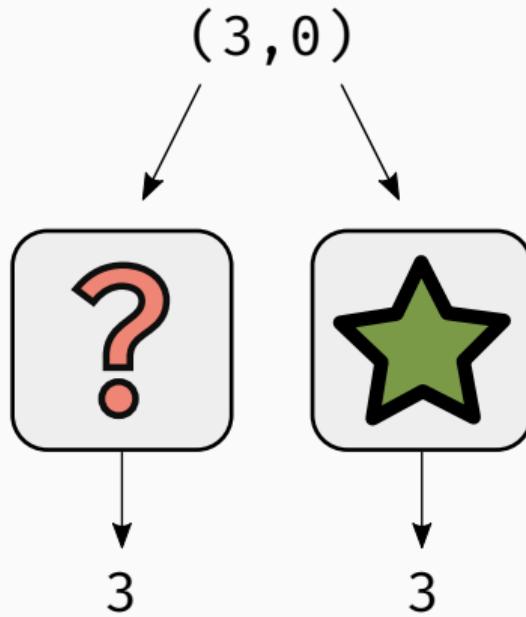


$\text{similarity}(4, 6) = 0.78$

$\text{similarity}(0, 3) = 0.33$

$\text{similarity}(3, 3) = 1.0$

Score Calculation



$$\text{similarity}(4, 6) = 0.78$$

$$\text{similarity}(0, 3) = 0.33$$

$$\text{similarity}(3, 3) = 1.0$$

average score: 0.70

Output Similarity: $\text{similarity}(O, O')$

11110111100100001000110010000000

11100010000110011110101100000000

Let's compare:

Output Similarity: $\text{similarity}(O, O')$

111	1011110010000100011001	00000000
111	00010000110011110110110	00000000

Are they in the same range?

Output Similarity: $\text{similarity}(O, O')$

1	1	1	1	0	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0
1	1	1	0	0	0	1	0	0	0	1	1	1	1	0	1	0	1	1	0

How many bits are different?

Output Similarity: $\text{similarity}(O, O')$

11110111100100001000110010000000
00010101011101101010000110000000
11100010000110011110101100000000

How close are they numerically?

DEMO

How to synthesize obfuscated code?

Obtaining Code



static disassembly

Obtaining Code



static disassembly

```
54 68 69 73 20 64 6f  
65 73 6e 27 74 20 6c  
6f 6f 6b 20 6c 69 6b  
65 20 61 6e 79 74 68  
69 6e 67 20 74 6f 20  
6d 65 2e de ad be ef
```

memory dump

Obtaining Code



static disassembly

```
54 68 69 73 20 64 6f  
65 73 6e 27 74 20 6c  
6f 6f 6b 20 6c 69 6b  
65 20 61 6e 79 74 68  
69 6e 67 20 74 6f 20  
6d 65 2e de ad be ef
```

memory dump

```
mov r15, 0x200  
xor r15, 0x800  
mov rbx, rbp  
add rbx, 0xc0  
mov rbx, qword ptr [rbx]  
mov r13, 1  
mov rcx, 0  
mov r15, rbp  
add r15, 0xc0  
or rcx, 0x88  
add rbx, 0xb  
mov r15, word ptr [r15]  
or r12, 0xffffffff80000000  
sub rcx, 0x78  
movzx r10, word ptr [rbx]  
xor r12, r13  
add r12, 0xffff  
add r15, 0  
mov r8, rbp  
sub rcx, 0x10  
or r12, r12  
or rcx, 0x800  
movzx r11, word ptr [r15]  
xor rcx, 0x800  
mov r12, r15  
add r8, 0  
xor r12, 0xf0  
mov rbx, 0x58  
add r11, rbp  
mov r15, rdx  
xor r10d, dword ptr [r12]  
sub r15, 0x800  
or rdx, 0x400  
mov rsi, 0x200  
mov r14, rbp  
sub rsi, rsi  
mov rdi, rbp  
sub r8, 0x400  
rst, r9  
sub r8, rsi  
add r14, 0  
add rsi, rax  
and r8, 0x80  
rst, r14  
mov r11, rbp  
add rdi, 0x80  
sub r8, rdi  
add r12, 0x78  
add rsi, r11  
mov rcx, 0x200  
mov rdi, qword ptr [rdi]  
dword ptr [rsi], 0x254  
xor rcx, 0xf0  
add rcx, r10  
add rdi, 6  
mov r8, 0x400  
mov ax, word ptr [rdi]  
r8, 1
```

instruction trace

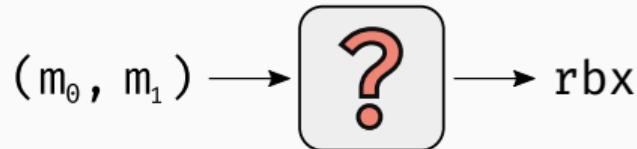
Learning Code Semantics

```
__handle_vnor:  
    mov    rcx, [rbp]  
    mov    rbx, [rbp + 4]  
    not    rcx  
    not    rbx  
    and    rcx, rbx  
    mov    [rbp + 4], rcx  
    pushf  
    pop    [rbp]  
    jmp    __vm_dispatcher
```

Handler performing **nor**
(with flag side-effects)

Learning Code Semantics

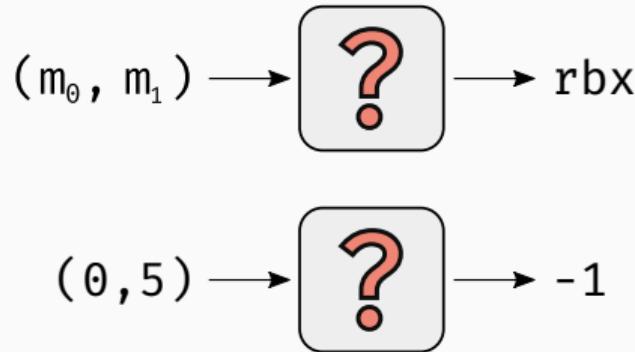
```
__handle_vnor:  
    mov    rcx, [rbp]  
    mov    rbx, [rbp + 4]  
    not    rcx  
• not    rbx  
    and    rcx, rbx  
    mov    [rbp + 4], rcx  
    pushf  
    pop    [rbp]  
    jmp    __vm_dispatcher
```



Handler performing **nor**
(with flag side-effects)

Learning Code Semantics

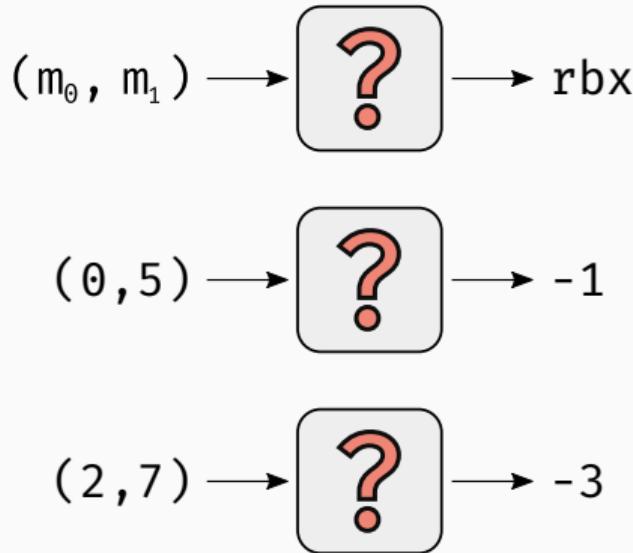
```
__handle_vnor:  
    mov    rcx, [rbp]  
    mov    rbx, [rbp + 4]  
    not    rcx  
• not    rbx  
    and    rcx, rbx  
    mov    [rbp + 4], rcx  
    pushf  
    pop    [rbp]  
    jmp    __vm_dispatcher
```



Handler performing **nor**
(with flag side-effects)

Learning Code Semantics

```
__handle_vnor:  
    mov    rcx, [rbp]  
    mov    rbx, [rbp + 4]  
    not    rcx  
• not    rbx  
    and    rcx, rbx  
    mov    [rbp + 4], rcx  
    pushf  
    pop    [rbp]  
    jmp    __vm_dispatcher
```

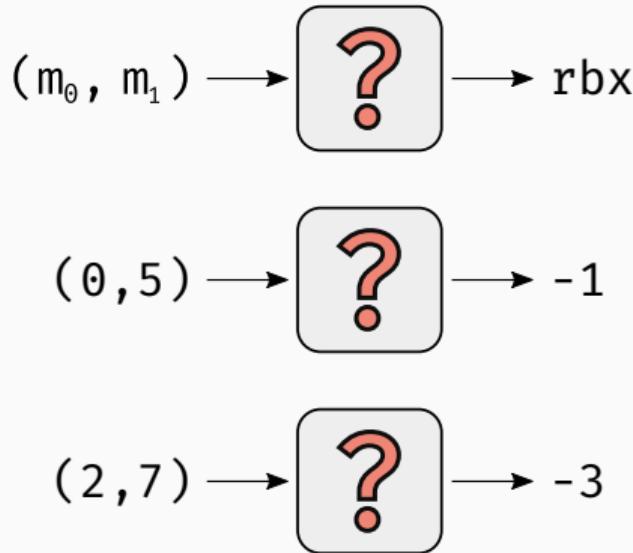


Handler performing **nor**
(with flag side-effects)

Learning Code Semantics

```
__handle_vnor:  
    mov    rcx, [rbp]  
    mov    rbx, [rbp + 4]  
    not    rcx  
• not    rbx  
    and    rcx, rbx  
    mov    [rbp + 4], rcx  
    pushf  
    pop    [rbp]  
    jmp    __vm_dispatcher
```

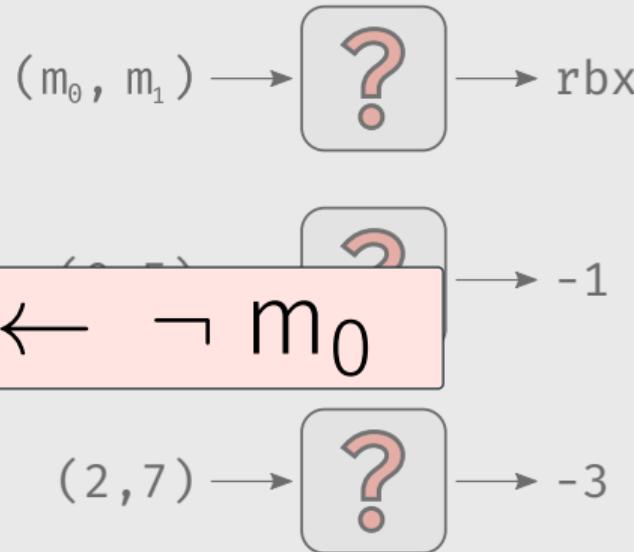
Handler performing **nor**
(with flag side-effects)



• • •

Learning Code Semantics

```
__handle_vnor:  
    mov    rcx, [rbp]  
    mov    rbx, [rbp + 4]  
    not    rcx  
• not    rbx  
    and    rcx, rbx  
    mov    [rbp + 4], rcx  
    pushf  
    pop    [rbp]  
    jmp    __vm_dispatcher
```

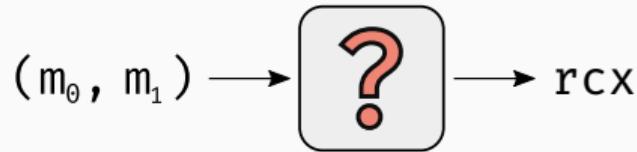


Handler performing nor
(with flag side-effects)

• • •

Learning Code Semantics

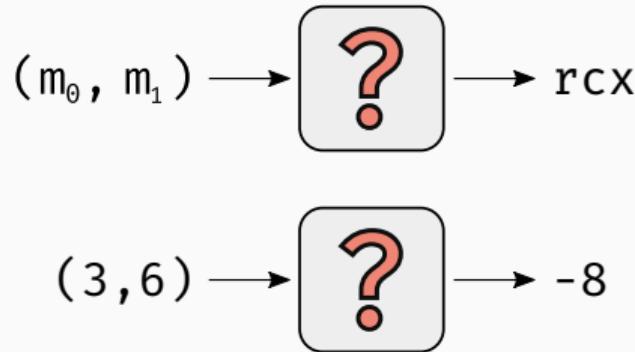
```
__handle_vnor:  
    mov    rcx, [rbp]  
    mov    rbx, [rbp + 4]  
    not    rcx  
    not    rbx  
• and    rcx, rbx  
    mov    [rbp + 4], rcx  
    pushf  
    pop    [rbp]  
    jmp    __vm_dispatcher
```



Handler performing `nor`
(with flag side-effects)

Learning Code Semantics

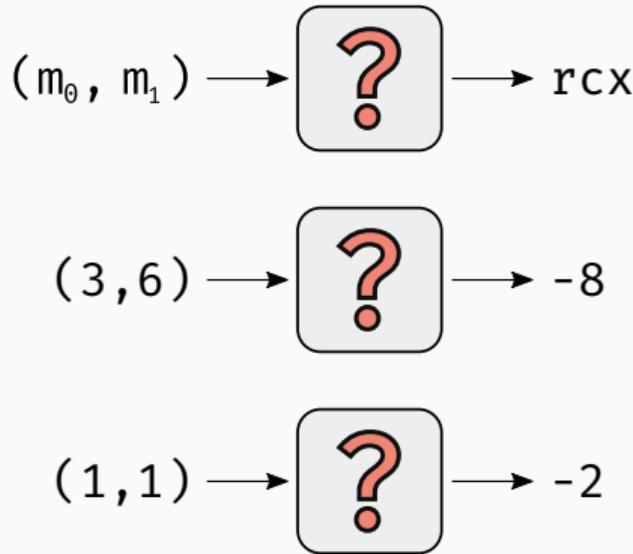
```
__handle_vnor:  
    mov    rcx, [rbp]  
    mov    rbx, [rbp + 4]  
    not    rcx  
    not    rbx  
• and    rcx, rbx  
    mov    [rbp + 4], rcx  
    pushf  
    pop    [rbp]  
    jmp    __vm_dispatcher
```



Handler performing **nor**
(with flag side-effects)

Learning Code Semantics

```
__handle_vnor:  
    mov    rcx, [rbp]  
    mov    rbx, [rbp + 4]  
    not    rcx  
    not    rbx  
• and    rcx, rbx  
    mov    [rbp + 4], rcx  
    pushf  
    pop    [rbp]  
    jmp    __vm_dispatcher
```

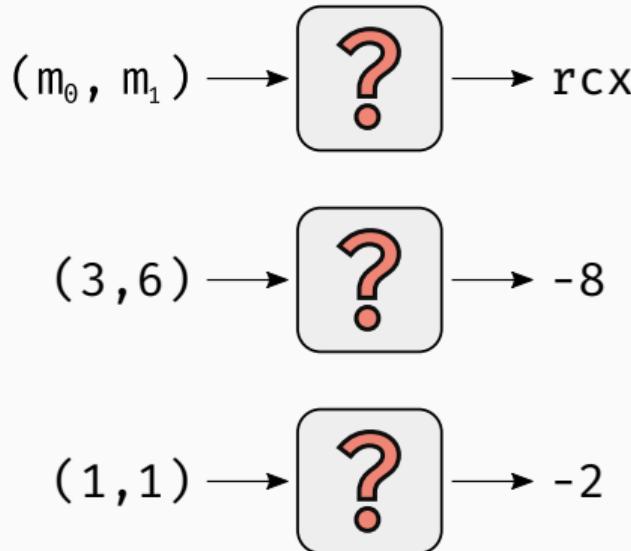


Handler performing nor
(with flag side-effects)

Learning Code Semantics

```
__handle_vnor:  
    mov    rcx, [rbp]  
    mov    rbx, [rbp + 4]  
    not    rcx  
    not    rbx  
• and    rcx, rbx  
    mov    [rbp + 4], rcx  
    pushf  
    pop    [rbp]  
    jmp    __vm_dispatcher
```

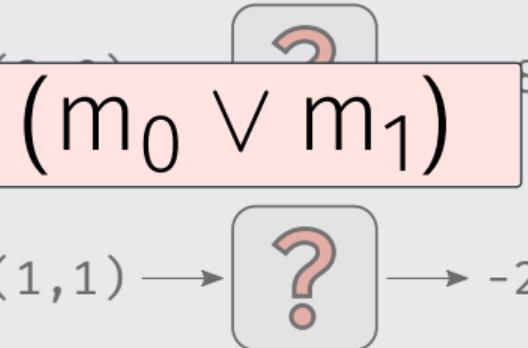
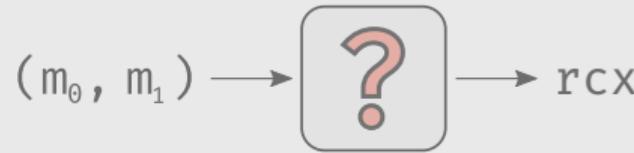
Handler performing **nor**
(with flag side-effects)



• • •

Learning Code Semantics

```
__handle_vnor:  
    mov    rcx, [rbp]  
    mov    rbx, [rbp + 4]  
    not    rcx  
    not    rbx  
• and   rcx, rbx  
    mov    [rbp + 4], rcx  
    pushf  
    pop    [rbp]  
    jmp    __vm_dispatcher
```

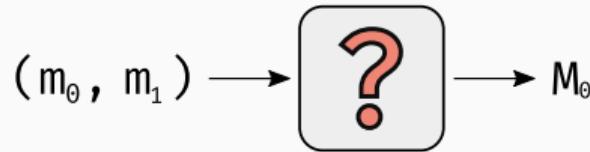


Handler performing nor
(with flag side-effects)

• • •

Learning Code Semantics

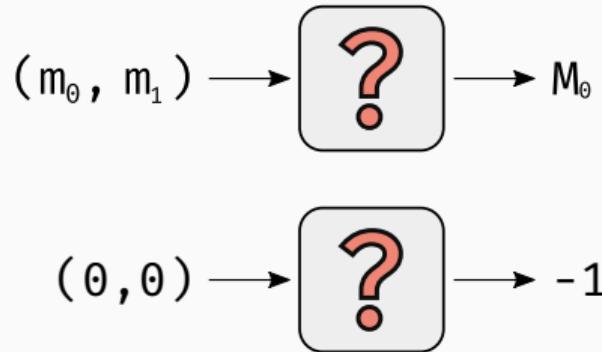
```
__handle_vnor:  
    mov    rcx, [rbp]  
    mov    rbx, [rbp + 4]  
    not    rcx  
    not    rbx  
    and    rcx, rbx  
• mov    [rbp + 4], rcx  
    pushf  
    pop    [rbp]  
    jmp    __vm_dispatcher
```



Handler performing **nor**
(with flag side-effects)

Learning Code Semantics

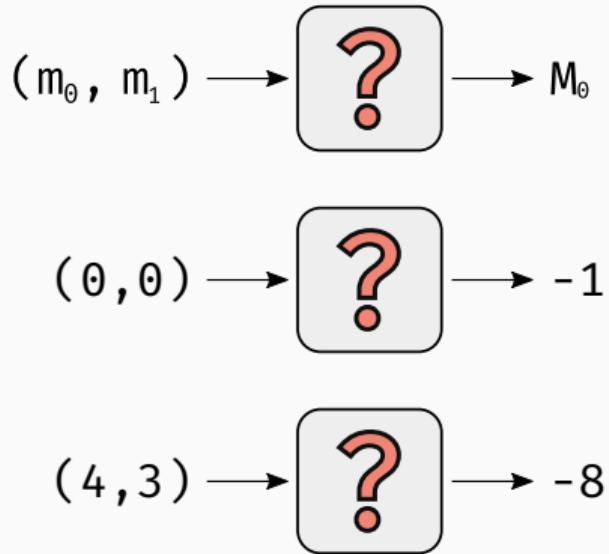
```
__handle_vnor:  
    mov    rcx, [rbp]  
    mov    rbx, [rbp + 4]  
    not    rcx  
    not    rbx  
    and    rcx, rbx  
•   mov    [rbp + 4], rcx  
    pushf  
    pop    [rbp]  
    jmp    __vm_dispatcher
```



Handler performing **nor**
(with flag side-effects)

Learning Code Semantics

```
__handle_vnor:  
    mov    rcx, [rbp]  
    mov    rbx, [rbp + 4]  
    not    rcx  
    not    rbx  
    and    rcx, rbx  
•   mov    [rbp + 4], rcx  
    pushf  
    pop    [rbp]  
    jmp    __vm_dispatcher
```

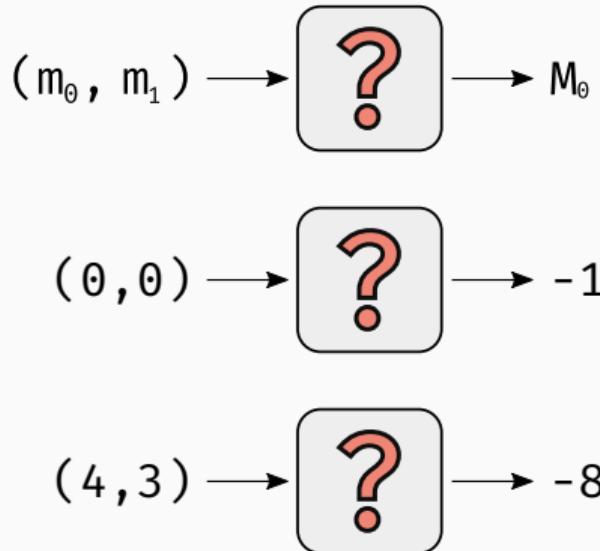


Handler performing **nor**
(with flag side-effects)

Learning Code Semantics

```
__handle_vnor:  
    mov    rcx, [rbp]  
    mov    rbx, [rbp + 4]  
    not    rcx  
    not    rbx  
    and    rcx, rbx  
•   mov    [rbp + 4], rcx  
    pushf  
    pop    [rbp]  
    jmp    __vm_dispatcher
```

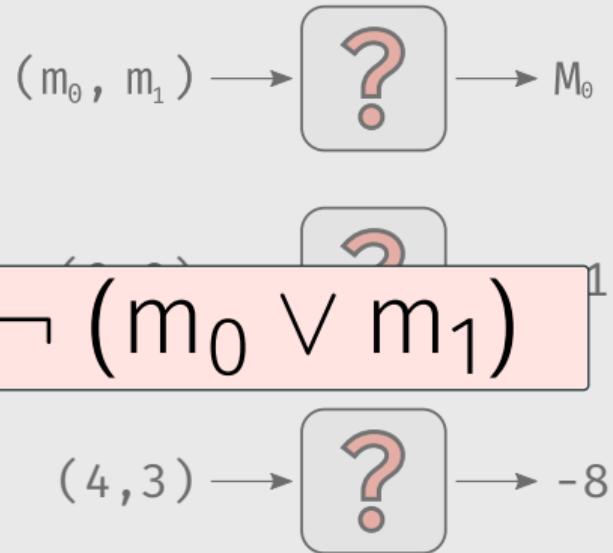
Handler performing **nor**
(with flag side-effects)



• • •

Learning Code Semantics

```
__handle_vnor:  
    mov    rcx, [rbp]  
    mov    rbx, [rbp + 4]  
    not    rcx  
    not    rbx  
    and    rcx, rbx  
•   mov    [rbp + 4], rcx  
    pushf  
    pop    [rbp]  
    jmp    __vm_dispatcher
```



Handler performing nor
(with flag side-effects)

• • •

Learning Code Semantics

```
__handle_vnor:  
    mov    rcx, [rbp]  
    mov    rbx, [rbp + 4]  
    not    rcx  
• not    rbx  
• and    rcx, rbx  
• mov    [rbp + 4], rcx  
    pushf  
    pop    [rbp]  
    jmp    __vm_dispatcher
```

$$\text{rbx} \leftarrow \neg m_0$$

$$\text{rcx} \leftarrow \neg (m_0 \vee m_1)$$

$$M_0 \leftarrow \neg (m_0 \vee m_1)$$

Handler performing **nor**
(with flag side-effects)

I/O Sampling

WinDbg



Valgrind

x64dbg



Unicorn

DynamoRIO



angr

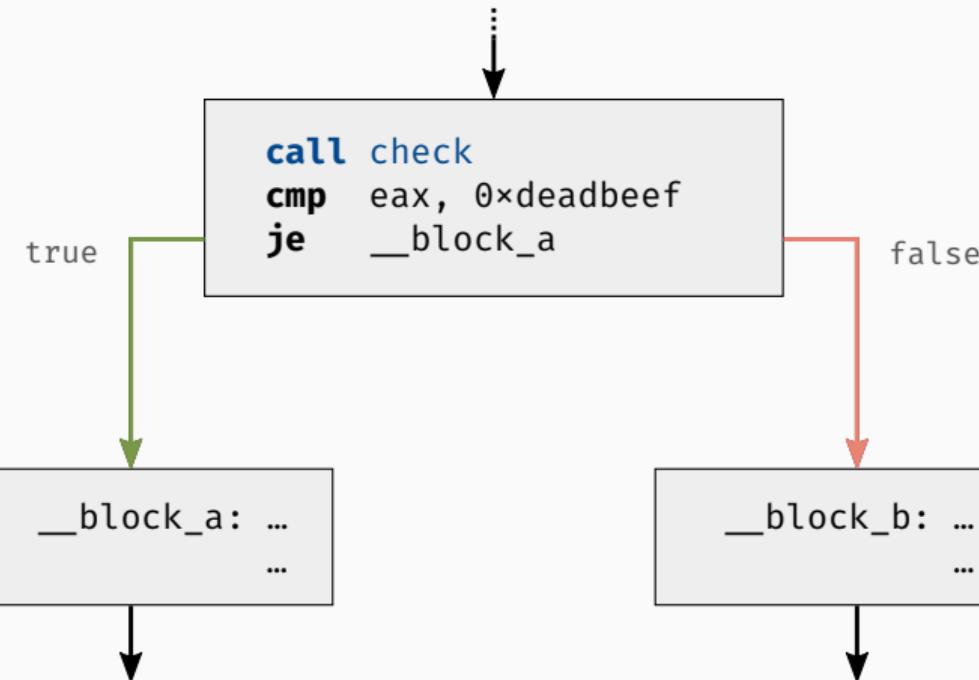


<your tool here>

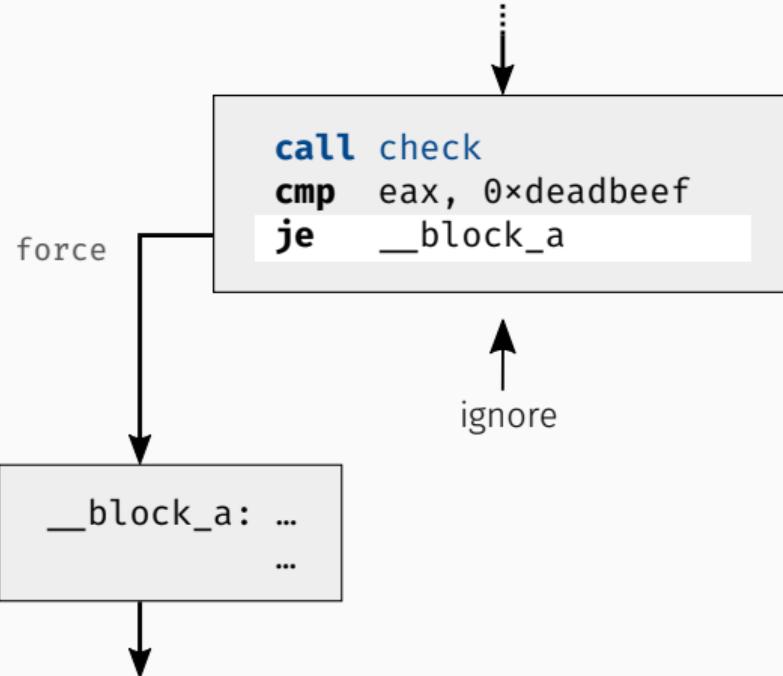
Miasm

Metasm

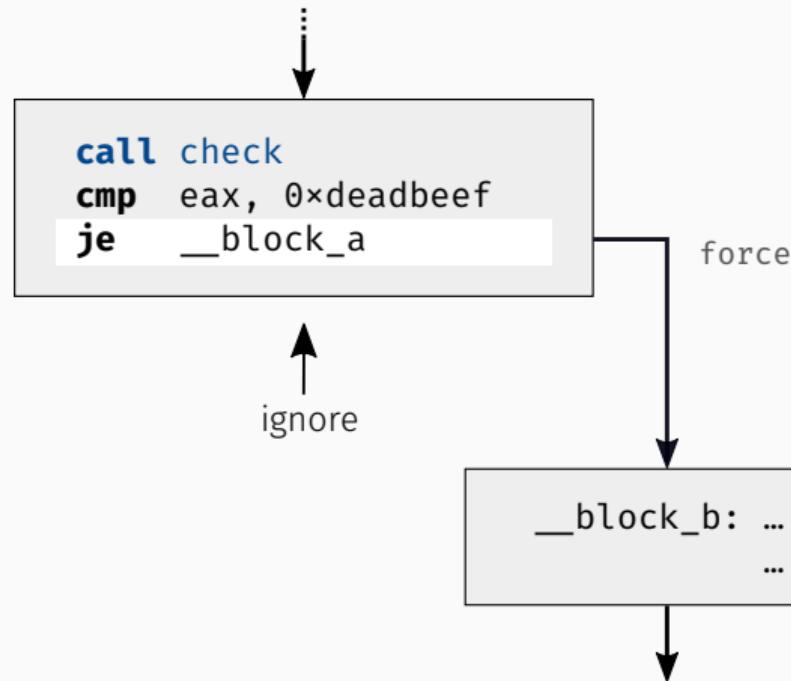
Instruction Trace: Forced Execution



Instruction Trace: Forced Execution



Instruction Trace: Forced Execution



Syntia

- program synthesis framework for code deobfuscation
- written in Python
- random I/O sampling for assembly code
- MCTS-based program synthesis

<https://github.com/RUB-SysSec/syntia>

DEMO

Breaking Virtual Machine Obfuscation

Reminder: Virtual Machine Hardening

Hardening Technique #1 – Obfuscating individual VM components.

Hardening Technique #2 – Duplicating VM handlers.

Hardening Technique #3 – No central VM dispatcher.

Hardening Technique #4 – No explicit handler table.

Hardening Technique #5 – Blinding VM bytecode.

#1: Obfuscating Individual VM Components

```
mov    r15, 0x200          mov    r15, rdx          add    r8, 1           or     r14, r14          mov    r14, 0x200          add    r15, 0x3f
xor    r15, 0x800          xor    r10d, dword ptr [r12]  or     r8, 0x78          mov    rax, rbp          rdx, 0xc0
mov    rbx, rbp          sub    r15, 0x800          add    word ptr [rbx], r10w  and    rcx, r13          and    r11, r14
add    rbx, 0xc0          or     rdx, 0x400          add    r15, rax          add    r11, 0x78          and    rsi, r9
mov    rbx, qword ptr [rbx] nov   r15, 0x200          add    rax, 4           or     r15, 0x88          add    rax, 0xc0
mov    r13, 1             nov   r14, rbp          sub    r15, rax          mov    rdx, qword ptr [rdx]  add    rdi, r14
mov    rcx, 0             nov   r14, rsi          pop    r9            add    r13, 0xffff          or     r15, 0x9f
mov    r15, rbp          sub    r15, rsi          mov    rcx, rbp          and    rcx, 0x20          add    rdx, 0xa
add    r15, 0xb0          nov   r15, rbp          add    r13, 0xc0          mov    r10, rbp          and    r11, 0x78
or     r15, 0x88          sub    r15, 0x400          add    r10, qword ptr [rcx]  mov    rbp, 0             and    rdi, 0x7fffffff
add    rbx, 0xb          sub    r15, r9           add    r13, 8            cmp    rbb, 0             add    rax, 0x58
add    r15, qword ptr [r15] nov   r14, rsi          add    r14, r8            je     0x49e          sub    rax, 0
or     r12, 0xffffffff80000000 add   r14, 0             mov    r10, 0x89          mov    rdx, rbp          or     rsi, 1
sub    r15, 0x78          add    r9, 0             xor    word ptr [r10], si  or     r11, 0x40          mov    rax, qword ptr [rax]
movzx r10, word ptr [rbx] xor    r14, r14          xor    rdx, r11          and    r15, 1           or     r9, rbp
and    r8, 0x88          add    r9, 0             mov    rsi, rbp          xor    r11, 0x10          mov    r13, 0x200
movzx r10, word ptr [rbx] xor    r14, r14          xor    r10, 0x10          and    rdx, rbx          mov    r10, 0x58
and    r8, 0x88          add    r9, 0             add    r13, 0xf0          add    rdx, 0xc0          mov    r15, 0
add    r12, 0xffff         xor    r14, r14          sub    r13, 0x20          and    rax, 0x40          or     r14, 4
add    r15, 0,               add    r9, 0             mov    rsi, 0             or     rbx, 0xf0          add    r9, 0
add    r15, 0,               sub    r13, 0x20          sub    r13, 0x20          mov    r15, 0x12           or     r10, 0x20
add    r8, rdi             add    r9, 0             add    r13, 0x20          xor    r15, qword ptr [rdx]  add    eax, dword ptr [r9]
sub    r15, 0x10             add    r9, 0             add    rsi, 0x5a          mov    rdx, 0x40           xor    r10, 0x40
add    r8, rbp             add    r9, 0             mov    r8, rcx          sub    r11, 0x8             add    eax, 0x3f505c07
or     r12, r12             add    r9, 0             mov    rax, word ptr [rsi]  add    rdx, 4
or     r15, 0x800           add    r9, 0             add    r14, 0             or     r11, 0x80           add    r15, 0x88
add    r15, 0x800           add    r9, 0             mov    r8, 0x58           mov    r8w, word ptr [rdx]  add    r12, rbp
xor    r15, 0x800           add    r9, 0             and    rax, rdx          mov    r14, r8           or     rdi, 0x90
xor    r15, 0x800           add    r9, 0             mov    rbx, qword ptr [rbx]  add    r12, 0             add    r12, 0
add    r12, 0x15             add    r9, 0             and    rcx, 0x20          add    r14, 0x89          xor    rbx, 0x80
add    r8, 0                 add    r9, 0             sub    r13, 0x20          xor    r13, 4           add    rdi, 0x80
add    r12, 0x15             add    r9, 0             add    r14, 0x89          pop    r10           add    rdx, 0xf0
add    r8, 0                 add    r9, 0             or     rax, 0x40          xor    si, 0x7328          mov    r13, 0x400
add    r12, 0x15             add    r9, 0             add    r15, 0x10          mov    qword ptr [r8], r10  add    dword ptr [r12], eax
add    r8, 0x400             add    r9, 0             xor    r13, 0x80          xor    rsi, 0x88           add    rsi, r8
add    r12, 0x15             add    r9, 0             add    r15, 0x10          xor    r8, 0x58           and    r10, 0
add    r8, 0x400             add    r9, 0             xor    r14, rbp          add    r14, 0x78           and    rbx, 0x20
add    r12, 0x15             add    r9, 0             add    r15, 0x10          mov    r10b, 0x68          and    rax, 0xffff
add    r8, 0x400             add    r9, 0             xor    r14, rbp          and    r9, 0x12           mov    r11, 0
add    r12, 0x15             add    r9, 0             add    r15, 0x10          or     rbi, r10           add    r13, r8
add    r8, 0x400             add    r9, 0             xor    r14, rbp          and    r15, 0x78           or     rbx, 1
add    r12, 0x15             add    r9, 0             add    r15, 0x10          mov    r14, rbp           shl    rax, 3
add    r8, 0x400             add    r9, 0             xor    r14, r12          or     r9, 0             add    r8, rax
add    r12, 0x15             add    r9, 0             add    r15, 0x10          add    r14, 0x29          or     rbx, r15
add    r8, 0x400             add    r9, 0             xor    r14, rbp          add    r14, 0x82           sub    r15, 0x10
add    r12, 0x15             add    r9, 0             add    r15, 0x10          xor    rbi, rdi          or     r11, r13
add    r8, 0x400             add    r9, 0             xor    r14, rbp          add    r15, 0x3f           mov    rbx, qword ptr [r8]
add    r12, 0x15             add    r9, 0             add    r15, 0x10          or     byte ptr [r14], r10b  add    rdx, rbp
add    r8, 0x400             add    r9, 0             xor    r11, r14          mov    rax, 0x58           sub    r13, 0x80
add    r12, 0x15             add    r9, 0             add    r15, 0x10          mov    r8, rbp           add    rdx, 0xc0
add    r8, 0x400             add    r9, 0             xor    r11, r14          sub    r11, 0x78           add    qword ptr [rdx], 0xd
add    r12, 0x15             add    r9, 0             add    r15, 0x10          add    r8, 0             xor    rbb, 0x3f
add    r8, 0x400             add    r9, 0             xor    r11, r14          add    r12, 0x88           mov    r8, qword ptr [r8]
add    r12, 0x15             add    r9, 0             add    r15, 0x10          xor    r12, 0x40           add    rdx, rbp
add    r8, 0x400             add    r9, 0             xor    r11, r14          add    r13, 1             jmp    rbx
add    r12, 0x15             add    r9, 0             add    r15, 0x10          xor    rsi, 1
add    r8, 0x400             add    r9, 0             xor    r11, r14          xor    rax, rbp
```

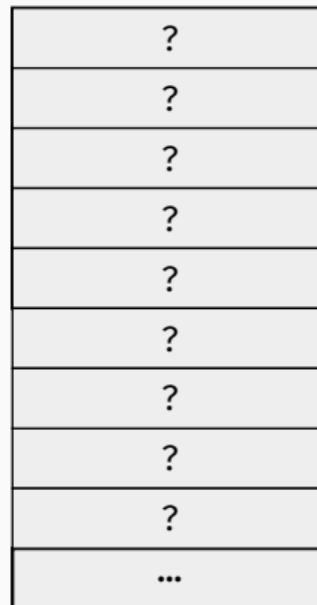
#1: Obfuscating Individual VM Components

```

mov    r15, 0x200          mov    r15, rdx          add    r8, 1           or     r14, r14        mov    r14, 0x200          add    r15, 0x3f
xor    r15, 0x800          xor    r10d, dword ptr [r12]  or     r8, 0x78        mov    rax, rbp        or     r15, 0xffffffff80000000
mov    rbx, rbp          sub    r15, 0x800          add    word ptr [rbx], r10w   and    rcx, r13        add    rdx, 0xc0
add    rbx, 0xc0          or     rdx, 0x400          add    r15, rax        add    r11, r14        and    rsi, r9
mov    rbx, qword ptr [rbx] nov   r15, 0x200          sub    r15, 4           or     r15, 0x88        add    rax, 0xc0
mov    r13, 1             nov   r14, rbp          pop    r9            add    r15, qword ptr [rdx]  add    rdi, r14
mov    rcx, 0             sub    rsi, rsi          add    r13, 0xfffff      add    rdx, 0xa         or     rsi, 1
mov    r15, rbp          mov    rsi, rbp          mov    rcx, rbp        and    r11, 0x78        mov    rax, qword ptr [rax]
add    r15, rbp          nov   rdi, rbp          add    r15, 0xc0        mov    r16, rbp        and    rdi, 0x7fffffff
or     r15, 0x800          mov    r15, 0x400          add    r13, r15        cmp    r8b, 0          add    rax, 0x58
or     r15, 0x88           sub    rsi, r9           add    r14, r8           je     0x49e        sub    rsi, 4
add    r15, 0xb8           sub    r8, rsi           add    r10, 0x89        mov    rdx, rbp        or     rbx, rsi
add    r15, qword ptr [r15] add    r14, 0             xor    word ptr [r10], si  or     r11, 0x40        movzx rax, word ptr [rax]
or     r12, 0xffffffff80000000 add   rsi, rax          add    r9, 0           xor    rdx, r11        mov    r9, rbp
sub    r10, 0x78           and    r8, 0x88          xor    r10d, dword ptr [r9]  mov    r11, 0x10        mov    r13, 0x200
movzx r10, word ptr [rbx] xor    r14, 0             mov    rsi, rbp        xor    r10, 0x58
add    r12, 0x10           xor    r13, r14          and    rdi, 0xffffffff80000000  add    r9, 0
add    r15, 0xffff          add    rsi, 0x0c0        sub    rdx, rbx        or     r10, 0x20
add    r15, 0               sub    r13, rdi          and    rax, 0x40        add    rdx, 0xc0
add    r15, 0               sub    r13, 0x20        or     rbx, 0xf0        mov    r15, 0x12
add    r15, 0               sub    r13, 0x28        add    rsi, 0x5a        mov    rdx, qword ptr [rdx]
add    r15, 0               or     r13, 0x88        xor    r8, rcx        sub    r11, r8        or     r10, 0x20
or     r12, r12             add    r14, 0             movzx rsi, word ptr [rsi]  add    rdx, 4        add    eax, dword ptr [r9]
or     r10, 0x800           and    r13, 8           mov    rax, 0x200        or     r11, 0x80        xor    r10, 0x40
or     r10, 0x800           mov    rdi, qwner        and    rdx, 4        add    r15, 0x5f505c07
movzx r11, word ptr [r15] add    r14, 0             mov    r10, 0x200        or     r11, 0x88        add    r12, rbp
xor    r10, 0x800           xor    r15, 0             mov    r10, 0x200        and    r12, 0x90
and    r12, r15             add    r14, 0             mov    r10, 0x200        or     rdi, 0x80
add    r8, 0                 add    r15, 0             mov    r10, 0x200        add    r12, 0
add    r12, 0xf8             nov   r8, r8           mov    r10, 0x200        or     rbx, 0x80
add    r12, 0xf8             nov   r14, rsi          add    r15, 0x20        add    rdi, 0x80
add    r12, 0xf8             nov   r15, rsi          sub    r11, r13        mov    r13, 0x400
add    r12, 0xf8             nov   r15, rsi          mov    r14, rbp        add    dword ptr [r12], eax
add    r11, rbp             nov   r15, rsi          add    r15, 0x40        add    rsi, r8
add    r11, rbp             nov   r15, rsi          add    rdx, 0x40        and    r10, 0
add    r11, rbp             nov   r15, rsi          add    rdx, 0x40        and    rbx, 0x20
add    r11, rbp             nov   r15, rsi          add    rdx, 0x40        and    rax, 0xffff
add    r11, rbp             nov   r15, rsi          add    rdx, 0x40        and    r11, 0
add    r11, rbp             nov   r15, rsi          add    rdx, 0x40        and    r13, r8
add    r11, rbp             nov   r15, rsi          add    rdx, 0x40        or     rbx, 1
add    r11, rbp             nov   r15, rsi          add    rdx, 0x40        shl    rax, 3
add    r11, rbp             nov   r15, rsi          add    rdx, 0x40        add    r8, rax
add    r11, rbp             nov   r15, rsi          add    rdx, 0x40        or     rbx, r15
add    r11, rbp             nov   r15, rsi          add    rdx, 0x40        sub    r15, 0x10
add    r11, rbp             nov   r15, rsi          add    rdx, 0x40        or     r11, r13
add    r11, rbp             nov   r15, rsi          add    rdx, 0x40        and    rbx, qword ptr [r8]
add    r11, rbp             nov   r15, rsi          add    rdx, 0x40        mov    rdx, rbp
add    r11, rbp             nov   r15, rsi          add    rdx, 0x40        sub    r13, 0x80
add    r11, rbp             nov   r15, rsi          add    rdx, 0x40        add    rdx, 0xc0
add    r11, rbp             nov   r15, rsi          add    rdx, 0x40        jmp    rbx
pushfq
xor    rbx, 0xf0           jbe    0x204          xor    esi, dword ptr [r9]  xor    rsi, 0x80
xor    rbx, 0x800           and    r8, r13          mov    r10, 0x8cc        xor    rbx, 0xffffffff80000000
and    rdx, r8             or     rcx, r13          sub    r15, 0x20        add    r10b, 0x68
and    rdx, r8             or     rcx, 4           xor    r9, 0           and    rax, 0xffff
and    rdx, r8             or     rdx, rbp          mov    r12, 0x58        xor    r11, 0
and    rdx, r8             or     rdx, rbp          xor    r9, 1           add    r13, r8
and    rdx, r8             or     rdx, rbp          add    r10, 0x12        or     rbx, 1
and    rdx, r8             or     rdx, rbp          xor    r10, r10        shl    rax, 3
and    rdx, r8             or     rdx, rbp          add    r11, r14        add    r8, rax
and    rdx, r8             or     rdx, rbp          add    r12, r15        or     rbx, r15
and    rdx, r8             or     rdx, rbp          add    r13, r15        sub    r15, 0x10
and    rdx, r8             or     rdx, rbp          add    r14, r15        or     r11, r13
and    rdx, r8             or     rdx, rbp          add    r15, r15        and    rbx, qword ptr [r8]
and    rdx, r8             or     rdx, rbp          add    r16, r15        mov    rdx, rbp
and    rdx, r8             or     rdx, rbp          add    r17, r15        sub    r13, 0x80
and    rdx, r8             or     rdx, rbp          add    r18, r15        add    rdx, 0xc0
and    rdx, r8             or     rdx, rbp          add    r19, r15        jmp    rbx
and    rdx, r8             or     rdx, rbp          add    r20, r15        xor    r10, 0
and    rdx, r8             or     rdx, rbp          add    r21, r15        xor    r8, rax
and    rdx, r8             or     rdx, rbp          add    r22, r15        or     rbx, r10
and    rdx, r8             or     rdx, rbp          add    r23, r15        sub    r11, r13
and    rdx, r8             or     rdx, rbp          add    r24, r15        and    rbx, qword ptr [r8]
and    rdx, r8             or     rdx, rbp          add    r25, r15        mov    rdx, rbp
and    rdx, r8             or     rdx, rbp          add    r26, r15        sub    r13, 0x80
and    rdx, r8             or     rdx, rbp          add    r27, r15        add    rdx, 0xc0
and    rdx, r8             or     rdx, rbp          add    r28, r15        jmp    rbx
add    r11, 0x2549b044      sub    r10, 0x800        xor    r13, 0x90        xor    r10, 0
add    r11, 0x2549b044      sub    r10, 0x800        xor    r13, 0x90        xor    r11, r13
add    r11, 0x2549b044      sub    r10, 0x800        xor    r13, 0x90        and    rbx, qword ptr [r8]
add    r11, 0x2549b044      sub    r10, 0x800        xor    r13, 0x90        mov    rdx, rbp
add    r11, 0x2549b044      sub    r10, 0x800        xor    r13, 0x90        sub    r13, 0x80
add    r11, 0x2549b044      sub    r10, 0x800        xor    r13, 0x90        add    rdx, 0xc0
add    r11, 0x2549b044      sub    r10, 0x800        xor    r13, 0x90        xor    r10, 0
add    r11, 0x2549b044      sub    r10, 0x800        xor    r13, 0x90        xor    rsi, 1
add    r11, 0x2549b044      sub    r10, 0x800        xor    r13, 0x90        xor    rax, rbp
add    r11, 0x2549b044      sub    r10, 0x800        xor    r13, 0x90        xor    r10, 0
add    r11, 0x2549b044      sub    r10, 0x800        xor    r13, 0x90        xor    r11, r13
add    r11, 0x2549b044      sub    r10, 0x800        xor    r13, 0x90        and    rbx, qword ptr [r8]
add    r11, 0x2549b044      sub    r10, 0x800        xor    r13, 0x90        mov    rdx, rbp
add    r11, 0x2549b044      sub    r10, 0x800        xor    r13, 0x90        sub    r13, 0x80
add    r11, 0x2549b044      sub    r10, 0x800        xor    r13, 0x90        add    rdx, 0xc0
add    r11, 0x2549b044      sub    r10, 0x800        xor    r13, 0x90        jmp    rbx

```

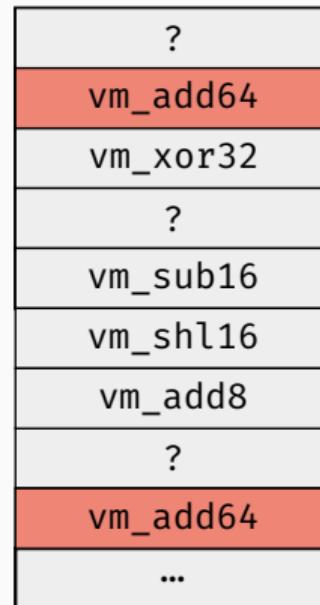
#2: Duplicating VM Handlers



#2: Duplicating VM Handlers

?
vm_add64
vm_xor32
?
vm_sub16
vm_shl16
vm_add8
?
vm_add64
...

#2: Duplicating VM Handlers



#5: Blinding VM Bytecode

```

mov    r15, 0x200          mov    r15, rdx          add    r8, 1           or    r14, r14          mov    r14, 0x200          add    r15, 0x3f
xor    r15, 0x800          xor    r10d, dword ptr [r12]  or    r8, 0x78          mov    rax, rbp          or    r15, 0xffffffff80000000
mov    rbx, rbp          sub    r15, 0x800          add    word ptr [rbx], r10w  and    rcx, r13          add    rdx, 0xc0
add    rbx, 0xc0          or    rdx, 0x400          add    r15, rax          add    rax, 4           and    r11, r14
mov    rbx, qword ptr [rbx] nov   r15, 0x200          sub    sub    r8, 0x80000000  or    r15, 0x88
mov    r13, 1             nov   r15, rbp          pop    r9            mov    rdx, qword ptr [rdx]  add    rdx, 0xa
mov    r15, 0               nov   rsi, rsi          mov    r15, 0xfffff        add    r11, 0x78
mov    r15, rbp          sub    rsi, rsi          mov    r10, rbp          mov    r8b, byte ptr [rdx]  and    rsi, 1
add    r15, 0xc0          nov   rdi, rbp          add    r10, 8            cmp    r8b, 0
or     r15, 0x88          nov   r15, 0x400          add    r13, r15          add    r11, 1
add    r15, 0x88          sub    rsi, r9            add    r14, r8            je    0x49e
add    r15, 0xb             sub    rsi, rsi          add    r10, 0x89          mov    rdx, rbp
mov    r15, qword ptr [r15] add    r14, 0             mov    r9, rbp          xor    word ptr [r10], si  add    r15, 0x3f
or     r12, 0xffffffff80000000 add    rsi, rax          add    r9, 0             or    r11, 0x40
sub    r15, 0x78          and    r10d, dword ptr [r9]  xor    rdx, r11          and    r15, 1
mov    movzx r10, word ptr [rbx] xor    r10d, 0xffffffff80000000  mov    rsi, rbp          and    rsi, r9
r10, word ptr [rbx] xor    rsi, r14          and    rdi, 0xfffffff        xor    r11, 0x10
mov    r12, r13             nov   rsi, rbp          sub    r13, 0xf0          xor    r10, 0x10
add    r12, 0xfffff         add    rdi, 0xc0          mov    rsi, 8            add    rdx, 0xc0
add    r15, 0               sub    r13, 0x20          add    rax, 0x40          or    r14, 4
add    r15, 0                sub    r13, 0x20          or    rbx, 0xf0          mov    r15, 0x12
add    r15, 0                sub    r13, 0x88          mov    r8, rcx          mov    rdx, qword ptr [rdx]
add    r15, 0                sub    r13, 0x10          add    r13, 0x5a          add    r11, 78
add    r15, 0                sub    r13, 0x80          xor    rax, 0x40          xor    sub    r1, r8
add    r15, 0                sub    r13, 0x80          or    rax, 0x40          add    eax, dword ptr [r9]
add    r15, 0                sub    r13, 0x80          xor    rax, 0x40          xor    r10, 0x40
add    r15, 0                sub    r13, 0x80          or    rax, 0x40          add    eax, 0x3f505c07
add    r15, 0                sub    r13, 0x80          xor    rax, 0x40          add    r15, 0x88
add    r15, 0                sub    r13, 0x80          or    rax, 0x80          mov    r12, rbp
add    r15, 0                sub    r13, 0x80          mov    r8w, word ptr [rdx]  or    r10, 0x58
add    r15, 0                sub    r13, 0x80          and    rax, rdx          mov    r14, r8
add    r15, 0                sub    r13, 0x80          mov    r8, rbp          add    r8, rbp
add    r15, 0                sub    r13, 0x80          add    r14, 0x80          or    r12, 0
add    r15, 0                sub    r13, 0x80          xor    r14, r8          add    r12, 0
add    r15, 0                sub    r13, 0x80          add    r14, 0x80          or    rbx, 0x80
add    r15, 0                sub    r13, 0x80          xor    r13, 4            add    rdi, 0xf0
add    r15, 0                sub    r13, 0x80          pop    r10            add    r13, 0x400
add    r15, 0                sub    r13, 0x80          xor    si, 0x7328        jmp    0x4ae
add    r15, 0                sub    r13, 0x80          mov    qword ptr [r8], r10  add    dword ptr [r12], eax
add    r15, 0                sub    r13, 0x80          add    rdx, 0x78          and    rsi, r9
add    r15, 0                sub    r13, 0x80          add    rdx, 0x20          xor    rsi, 0x88
add    r15, 0                sub    r13, 0x80          or    r9, 0xfffff        or    r10, 8
add    r15, 0                sub    r13, 0x80          sub    r9, 1            and    rbx, 0xffffffff80000000
add    r15, 0                sub    r13, 0x80          mov    r9, rbp          and    rax, 0x20
add    r15, 0                sub    r13, 0x80          add    rsi, 0x58          xor    r10b, 0x68
add    r15, 0                sub    r13, 0x80          xor    rax, rdx          xor    r9, 0x12
add    r15, 0                sub    r13, 0x80          add    r8, 0x80          mov    r11, 0
add    r15, 0                sub    r13, 0x80          mov    r15, rsi          add    r13, r8
add    r15, 0                sub    r13, 0x80          add    r14, rbp          xor    rbx, 1
add    r15, 0                sub    r13, 0x80          add    r8, r15          shl    rax, 3
add    r15, 0                sub    r13, 0x80          mov    rbx, 0            add    r8, rax
add    r15, 0                sub    r13, 0x80          and    rdx, 0x20          or    rbx, r15
add    r15, 0                sub    r13, 0x80          xor    r10, 0x10          sub    r15, 0x10
add    r15, 0                sub    r13, 0x80          add    r10, 0xcc          or    r11, r13
add    r15, 0                sub    r13, 0x80          xor    r11, r14          mov    rbx, qword ptr [r8]
add    r15, 0                sub    r13, 0x80          xor    r12, 0x14          mov    rdx, rbp
add    r15, 0                sub    r13, 0x80          add    r13, 0x10          sub    r13, 0x80
add    r15, 0                sub    r13, 0x80          mov    r12, 0x12          add    rdx, 0xc0
add    r15, 0                sub    r13, 0x80          add    r8, 0            add    qword ptr [rdx], 0xd
add    r15, 0                sub    r13, 0x80          xor    r11, r14          mov    rax, 0x58
add    r15, 0                sub    r13, 0x80          xor    r12, 0x14          mov    r8, rbp
add    r15, 0                sub    r13, 0x80          add    r13, 0x12          sub    rsi, 0x78
add    r15, 0                sub    r13, 0x80          mov    r12, 0x12          add    r8, 0x127
add    r15, 0                sub    r13, 0x80          add    r8, 0            xor    r8, 0x127
add    r15, 0                sub    r13, 0x80          xor    r14, 0x88          xor    rbx, 0x3f
add    r15, 0                sub    r13, 0x80          and    r14, 0x88          mov    r8, qword ptr [r8]
add    r15, 0                sub    r13, 0x80          add    r13, 1            xor    rsi, 1
add    r15, 0                sub    r13, 0x80          mov    rdx, rbp          mov    rax, rbp

```

#5: Blinding VM Bytecode

```

mov r15, 0x200          mov r15, rdx          add r8, 1           or r14, r14          mov r14, 0x200          add r15, 0x3f
xor r15, 0x800          xor r10d, dword ptr [r12]    or r8, 0x78        mov rax, rbp          or r15, 0xffffffff80000000
mov rbx, rbp            sub r15, 0x800         add word ptr [rbx], r10w    and rcx, r13        and r15, 0xc0
add rbx, 0xc0            or rdx, 0x4000       sub r15, rax        add rax, 4           and r11, r14        and rsi, 0
mov rbx, qword ptr [rbx] nov r15, 0x200       sub r8, -0x80000000    or r15, 0x88        and r11, r14        and rdi, r14
mov r13, 1               nov r15, rbp        mov r13, 0xfffff      mov rdx, 0xa        or r11, 0x78        or r1, 1
mov r15, rbp            sub r15, rsi        add r13, 0x20          mov r8b, byte ptr [rdx]    and r11, 0x20        and rsi, 1
add r15, 0xc0            nov r15, rbp        and r13, 0x20          mov r8b, 0           add rax, 2
or r15, 0x88             sub r15, rsi        add r11, 0x78        or rax, 0xc0
add r15, 0xb8            sub r15, rsi        sub r11, 0x20        and rdi, r14
mov r15, qword ptr [r15] add r14, 0           add r11, 0x78        or r1, 0
or r12, 0xffffffff80000000 add rsi, rax        add r11, 0x40        and rsi, 4
sub r15, 0x78            and r8, 0x88        and r11, 0x40        or r15, rbp
movzx r10, word ptr [rbx] xor r15, r14        xor r11, 0x10        mov r15, 0x3f
add r12, 0xffff          xor r15, rbp        add rdx, 0xc0
add r15, 0,                 add rdi, 0xc0        or r14, 4           mov r15, 0x20
mov r8, rbp              sub r8, rdi        mov r15, 0x12           mov r13, rbp
sub r15, 0x10              add r8, 0x78        mov rdx, qword ptr [rdx]
or r12, r12               nov r15, 4           sub r11, 78          sub r10, 0x40
or r15, 0x800              nov r15, rdx        add rdx, 4           add eax, dword ptr [r9]
mov r11, word ptr [r15]   add rdi, qword ptr [rdi]    or r11, 0x80        xor r10, 0x40
xor r15, 0x800            add dword ptr [rsi], 0x2549    mov r15, 0x12
xor r15, 0x15              add r15, r10        mov rdx, qword ptr [rdx]
add r8, 0                  add rdi, 6           sub r11, 8           add eax, 0x3f505c07
add r8, 0x58                add r8, r10        add rdx, 4           add r15, 0x88
add r11, rbp              nov r8, 0x4000       mov r11, 0x80           mov r12, rbp
add r11, 0x800              nov r8, word ptr [rdi]    mov r8w, word ptr [rdx]
and r12, 0x29                nov r8, r10        add r14, r8           or r12, 0x98
add rbx, 0x800              sub r15, 8           add r8, rbp           add r12, 0
add r11, qword ptr [r11]    add r15, r10        xor r13, 4           or r13, 0x80
add rbx, 1                  add rdi, 29         pop r10             add rdi, 0xf0
and r12, r9                  or r15, rsi        mov qword ptr [r8], r10
add r12, r9                  or r15, r14        jmp 0x4ae             mov r13, 0x400
add r12, 1                  or r15, rsi        xor rsi, 0x88         add dword ptr [r12], eax
and r12, r9                  or r15, r14        add r10b, 0x68        and rsi, 0
add r12, 0x20                or r15, r14        mov r9, 0x68           or r10, 0
add r10d, dword ptr [r8]    add r15, r10        or r15, 0x68        and rbx, 0x20
sub r9, r11                add r15, r10        and r10b, 0x68        and rax, 0xffff
pushfq                   cmp r13b, 0xd2        mov r15, rsi           or r11, 0
xor rbx, 0xf0                jbe 0x2000           and r15, 0x78        and rax, 0x20
xor rbx, 0x800                and r8, r13        add r14, rbp           or r11, 8
and rdx, r8                  or r15, r13        add r8, r15           and r13, r8
mov r12, rbp                or r15, r12        mov r14, rbp           or rbx, 1
xor rdx, 0x20                or r15, r12        sub r14, 8            shl rax, 3
add r11, 0x2549b044         or r15, r14        xor r14, r15          add r8, rax
or rbx, 0x78                add r15, r14        add r14, 0x29          or r15, r15
and rdx, r10                add r15, r14        xor r14, r15          sub r15, 0x10
mov rax, 0                  add r15, r14        add r15, 0x3f          or r11, r13
add r12, 0x42                add r15, r14        or r15, 0x3f          and rbx, 0x20
add r15, r15                add r15, r14        mov r14, r10           or r11, r13
add r15, r15                add r15, r14        sub r11, 0x10          and rdx, 0xc0
add r15, r15                add r15, r14        add r15, 0x12           mov rdx, 0x80
and r12, 0x80                add r15, r14        add r15, 0x127         add qword ptr [rdx], 0xd
mov r12, 1                  add r15, r14        mov r12, 0x127         mov rax, 0x58
add r15, r15                add r15, r14        xor r12, r12           or r15, 0x88
add r15, r15                add r15, r14        add r14, 0x88           and rdx, 0xc0
and r12, 0x80                add r15, r14        sub r11, 0x78           mov rdx, 0x80
mov r12, 1                  add r15, r14        add r14, 0x88           add r13, 0x80
add r15, r15                add r15, r14        xor r12, 1             mov r8, qword ptr [r8]
add r15, r15                add r15, r14        mov r13, 1             xor rsi, 1
and r12, 0x80                add r15, r14        mov rdx, rbp           mov rax, rbp

```

The diagram illustrates the flow of control and data through the VM bytecode. A central box highlights the assembly code for the stack frame, specifically the stack slot for variable `r9`. The code within this box includes:

- `mov r9, rbp`
- `...`
- `add r9, 0`
- `...`
- `add eax, dword ptr [r9]`
- `...`
- `add eax, 0x3f505c07`
- `...`
- `mov r12, rbp`
- `...`
- `add r12, 0`
- `dword ptr [r12], eax`

Arrows point from the highlighted code to the surrounding assembly, showing how values are loaded from memory, modified, and stored back. The stack frame is identified by the label `[r14]` above the highlighted area.

#5: Blinding VM Bytecode

```

mov r15, 0x200      mov r15, rdx      add r8, 1          or r14, r14      mov r14, 0x200      add r15, 0x3f
xor r15, 0x800      xor r10d, dword ptr [r12]  or r8, 0x78       mov rax, rbp      or r15, 0xffffffff
mov rbx, rbp        sub r15, 0x800      add word ptr [rbx], r10w  and rcx, r13      and r15, 0xc0
add rbx, 0xc0        or rdx, 0x400      sub r15, rax      add rax, 4       or r15, 0x88
mov rbx, qword ptr [rbx] nov rsi, 0x200    sub r8, 0x80000000  and r15, 0xfffff
mov r13, 1           mov r15, rbp      mov r15, 0x80000000  mov rdx, qword ptr [rdx]
mov rcx, 0           sub rsi, rbp      add r13, 0xfffff   add rdx, 0xa
mov r15, rbp        nov rsi, rsi      and r15, 0x20      mov r8b, byte ptr [rdx]
add r15, 0xc0        mov rdi, rbp      or r11, 0x78      or r11, 0x78
or rcx, 0x88         sub rsi, r9       add r13, 1       mov r8b, 0
add rbx, 0xb8        sub rsi, rsi      and r15, 1       sub rsi, 4
add rbx, 0xb8        mov r9, r9      add r11, 0x78      or r8b, rsi
mov r15, qword ptr [r15] add r14, 0      mov r11, 0x78     movzx r9, word ptr [rax]
or r12, 0xfffffffff add rsi, rax      add rdx, 0x8a     and rdi, 0x7fffffff
sub rcx, 0x78        add r8, 0x88      add r11, 1       add rax, 0xc0
movzx r10, word ptr [rbx] xor rsi, r14    add r11, 0x14     and rsi, 1
xor r12, r13        mov rsi, rbp      add r11, 0x40     or r8b, rsi
add r12, 0xfffff    add rdi, 0xc0      add r11, 0x40     mov r15, rbp
add r15, 0           sub r8, rdi      add r11, 0x40     or r15, 0x200
mov r8, rbp        add r8, 0x78      add r11, 0x40     and r10, 0x58
sub rcx, 0x10        add rsi, 4       add r11, 0x40     add r9, 0
or r12, r12         nov rcx, 0x200    add r11, 0x40     or r10, 0x20
or rcx, 0x800        mov rdi, qword ptr [rdi]  add r11, 0x40     add eax, dword ptr [r9]
movzx r11, word ptr [r15] add dword ptr [rsi], 0x2549  add r11, 0x40     xor r10, 0x40
xor rcx, 0x800      xor r9, 0x2fa    add r11, 0x40     add eax, 0x3f505c07
add r12, r15        add r9, 0x2fa    add r11, 0x40     or r11, 0x88
add r8, 0           add r9, 0x2fa    add r11, 0x40     mov r12, rbp
add r12, 0x80       add r9, 0x2fa    add r11, 0x40     or rdi, 0x98
add rbx, 0x58       add r9, 0x2fa    add r11, 0x40     add r12, 0
add r11, rbp        add r9, 0x2fa    add r11, 0x40     add r12, 0x88
xor rbx, 0x800      add r9, 0x2fa    add r11, 0x40     or r12, rbp
and r12, 0x29       add r9, 0x2fa    add r11, 0x40     or rdi, 0x98
add rbx, 0x800      add r9, 0x2fa    add r11, 0x40     add r12, 0
add r11, qword ptr [r11] add r9, 0x2fa    add r11, 0x40     add r12, 0x80
mov r11, rdi        add r9, 0x2fa    add r11, 0x40     or r12, rsi
add rbx, 1           add r9, 0x2fa    add r11, 0x40     add r13, r8
and r12, r9         add r9, 0x2fa    add r11, 0x40     or r13, r8
add rdx, 1           add r9, 0x2fa    add r11, 0x40     add r13, r8
xor r18d, dword ptr [r8] add r9, 0x2fa    add r11, 0x40     or r13, r8
sub r9, r11          add r9, 0x2fa    add r11, 0x40     add r14, 0x29
pushfq              add r9, 0x2fa    add r11, 0x40     add r14, 0x29
xor rbx, 0xf0        add r9, 0x2fa    add r11, 0x40     or r15, r15
xor rbe, 0x800       add r9, 0x2fa    add r11, 0x40     and rax, 0x20
and rdx, r8           add r9, 0x2fa    add r11, 0x40     and rax, 0xfffff
or r12, rbp          add r9, 0x2fa    add r11, 0x40     or r15, 0xc0
xor rdx, 0x20         add r9, 0x2fa    add r11, 0x40     and r15, 0x8
sub rbx, 4            add r9, 0x2fa    add r11, 0x40     add r13, r8
add r11, 0x2549b044  add r9, 0x2fa    add r11, 0x40     or r13, r8
or rbx, 0x78          add r9, 0x2fa    add r11, 0x40     add r13, r8
and rdx, r10          add r9, 0x2fa    add r11, 0x40     or r11, r13
mov rax, 0             add r9, 0x2fa    add r11, 0x40     add r13, r8
add r12, 0x42          add r9, 0x2fa    add r11, 0x40     or r11, r13
add r15, 0x3f

```

No influence on underlying code's semantics

#3: No Central VM Dispatcher

```
mov    r15, 0x200          mov    r15, rdx          add   r8, 1           or    r14, r14          mov    r14, 0x200          add   r15, 0x3f
xor    r15, 0x800          xor    r10d, dword ptr [r12]      or    r8, 0x78          mov    rax, rbp          or    r15, 0xffffffff80000000
mov    rbx, rbp          sub   r15, 0x800          add   word ptr [rbx], r10w      and   rcx, r13          add   rdx, 0xc0
add   rbx, 0xc0          or    rdx, 0x400          mov    r15, rax          add   rax, 4           and   r11, r14
mov    rbx, qword ptr [rbx]      nov   r15, 0x200          sub   r8, 0x80000000      or    r15, 0x88
mov    r13, 1             nov   r15, rbp          mov    r15, 0xfffffff       mov    rdx, qword ptr [rdx]
or    r15, 0x800          sub   r15, rsi          add   r13, 0xfffff      add   r11, 0x78
add   r15, rbp          mov    r15, rsi          add   r13, 0x80000000     mov    r8b, byte ptr [rdx]
or    r15, 0x800          add   r15, rbp          add   r14, r8           cmp   r8b, 0
add   r15, rbp          sub   r15, r9           je    0x849e          add   r11, 0x14
add   r15, rbp          mov    r10, word ptr [rcx]      mov    rdx, rbp          and   rsi, 1
add   r15, rbp          add   r15, r9           add   r10, 0x89          mov    rax, qword ptr [rax]
add   r15, rbp          mov    r9, rbp          xor   word ptr [r10], si      or    r15, 0x1
add   r15, rbp          add   r9, 0            or    r11, 0x40          and   rsi, 4
add   r15, 0x800          xor   r10d, dword ptr [r9]      xor   rdx, 0xc0
add   r15, 0x800          sub   r13, 0xf0          mov    rsi, rbp          xor   r11, 0x10
add   r15, 0x800          and   rdi, 0xfffffff00000000      xor   r13, 0x80000000      mov    r13, 0x200
add   r15, 0x800          sub   r13, 0xf0          mov    r13, 0x800          add   rdx, 0x58
add   r15, 0x800          and   rdi, 0x8c0          xor   r13, 0x800          or    r10, 0x58
add   r15, 0x800          sub   r13, 0xf0          mov    r13, 0x800          or    r9, 0
add   r15, 0x800          and   rdi, 0x8c0          xor   r13, 0x800          or    r10, 0x20
add   r15, 0x800          sub   r13, 0xf0          mov    r13, 0x800          mov    r9, rbp
add   r15, 0x800          and   rdi, 0x8c0          xor   r13, 0x800          xor   r13, 0x800
add   r15, 0x800          sub   r13, 0xf0          mov    r13, 0x800          mov    r10, 0x40
add   r15, 0x800          and   rdi, 0x8c0          xor   r13, 0x800          or    r15, 0x88
add   r15, 0x800          sub   r13, 0xf0          mov    r13, 0x800          add   r15, 0x88
add   r15, 0x800          and   rdi, 0x8c0          xor   r13, 0x800          mov    r12, rbp
add   r15, 0x800          sub   r13, 0xf0          mov    r13, 0x800          or    rdi, 0x98
add   r15, 0x800          and   rdi, 0x8c0          xor   r13, 0x800          add   r12, 0
add   r15, 0x800          sub   r13, 0xf0          mov    r13, 0x800          or    rbx, 0x88
add   r15, 0x800          and   rdi, 0x8c0          xor   r13, 0x800          add   rdi, 0xf0
add   r15, 0x800          sub   r13, 0xf0          mov    r13, 0x800          add   r13, 0x8
add   r15, 0x800          and   rdi, 0x8c0          xor   r13, 0x800          or    r13, 0x400
add   r15, 0x800          sub   r13, 0xf0          mov    r13, 0x800          add   dword ptr [r12], eax
add   r15, 0x800          and   rdi, 0x8c0          xor   r13, 0x800          and   rsi, 0
add   r15, 0x800          sub   r13, 0xf0          mov    r13, 0x800          or    r10, 0
add   r15, 0x800          and   rdi, 0x8c0          xor   r13, 0x800          and   rbx, 0x20
add   r15, 0x800          sub   r13, 0xf0          mov    r13, 0x800          and   rax, 0xfffff
add   r15, 0x800          and   rdi, 0x8c0          xor   r13, 0x800          mov    r11, 0
add   r15, 0x800          sub   r13, 0xf0          mov    r13, 0x800          add   r13, r8
add   r15, 0x800          and   rdi, 0x8c0          xor   r13, 0x800          or    rbx, 0
add   r15, 0x800          sub   r13, 0xf0          mov    r13, 0x800          shl   rax, 3
add   r15, 0x800          and   rdi, 0x8c0          xor   r13, 0x800          add   r8, rax
add   r15, 0x800          sub   r13, 0xf0          mov    r13, 0x800          or    rbx, r15
add   r15, 0x800          and   rdi, 0x8c0          xor   r13, 0x800          sub   r15, 0x10
add   r15, 0x800          sub   r13, 0xf0          mov    r13, 0x800          or    r11, r13
add   r15, 0x800          and   rdi, 0x8c0          xor   r13, 0x800          mov    r10, 0x80
add   r15, 0x800          sub   r13, 0xf0          mov    r13, 0x800          add   rdx, 0xc0
add   r15, 0x800          and   rdi, 0x8c0          xor   r13, 0x800          add   qword ptr [rdx], 0xd
add   r15, 0x800          sub   r13, 0xf0          mov    r13, 0x800          jmp   ebx
```

#3: No Central VM Dispatcher

```

mov    r15, 0x200          mov    r15, rdx          add   r8, 1           or    r14, r14
xor    r15, 0x800          xor    r10d, dword ptr [r12]  or    r8, 0x78        mov    rax, rbp
mov    rbx, rbp            sub    r15, 0x800        add   word ptr [rbx], r10w  and   rcx, r13
add   rbx, 0xc0             or    r15, rax        add   rax, 4           or    r15, 0x88
mov    rbx, qword ptr [rbx]  nov   r15, 0x200        sub   r8, -0x80000000  mov    rdx, qword ptr [rdx]
mov    r13, 1               nov   r15, rbp        add   r13, 0xfffff      and   rdx, 0xa
mov    r15, rbp            sub    r15, rsi        add   r11, 0x78        mov    r8b, byte ptr [rdx]
add   r15, 0xc0             nov   r15, rbp        and   r11, 0x20        cmp   r8b, 0
or    r15, 0x88             sub    r15, rsi        je    0x49e         mov    rax, qword ptr [rax]
add   r15, 0x88             sub    r15, rsi        mov    rdx, rbp        and   rsi, 2
or    r15, 0x88             sub    r15, rsi        or    r11, 0x40        and   rdi, 0x7fffffff
add   r15, 0x88             sub    r15, rsi        and   r15, 1           add   rax, 0xc0
mov    r15, qword ptr [r15]  add   r14, 0           xor   r11, 0x10        add   rdi, r14
or    r12, 0xffffffff80000000 add   rsi, rax        add   rdx, 0xc0        or    rsi, 1
sub   r15, 0x78             and   r14, 0           xor   r11, 0x10        and   rdi, 0x7fffffff
movzx r10, word ptr [rbx]  xor   r14, r14        add   r11, 0x78        mov    rax, qword ptr [rax]
xor   r12, r13              xor   r14, r14        and   r11, 0x20        or    r9, rbp
add   r12, 0xfffff          add   rsi, rbp        mov    r9, rbp        mov    r13, 0x200
add   r15, 0                sub   r15, rdi        or    r11, 0x40        mov    r10, 0x58
mov    r8, rbp              add   r15, 0xc0        and   r11, 0x10        add   r9, 0
add   r15, 0x10              add   r8, 0x78        add   r14, 4           or    r10, 0x20
sub   r15, 0x10              add   r15, 0x12        mov    r15, 0x12        add   eax, dword ptr [r9]
mov    r12, r12              nov   r15, r10        mov    rdx, qword ptr [rdx]  xor   r10, 0x40
or    r12, r12              nov   r15, r10        sub   r11, r8           add   eax, 0x3f505c07
add   r12, r12              nov   r15, r10        add   r1x, 4           add   r15, 0x88
or    r12, r12              nov   r15, r10        or    r11, 0x80         mov    r12, rbp
or    r15, 0x800             nov   r15, r10        word ptr [rdx]       mov    r13, 0x400
movzx r11, word ptr [r15]  nov   r15, r10        mov    r14, 0           add   rdi, 0x12, eax
add   r15, 0x800             nov   r15, r10        add   r14, 0           and   rsi, 8
mov    r15, 0x800             nov   r15, r10        mov    r15, 0x6f0        or    r10, 0
and   r15, 0x800             nov   r15, r10        add   r15, 0xf0        add   r12, 0
add   r15, 0x800             nov   r15, r10        xor   r15, 0x10        or    rbx, 0x80
add   r15, 0x800             nov   r15, r10        add   r15, 6           add   rdi, 0xf0
add   r15, 0x800             nov   r15, r10        pop   r10           mov    r13, 0x400
add   r15, 0x800             nov   r15, r10        mov    qword ptr [r10], r10  jnp   0x4ae
add   r15, 0x800             nov   r15, r10        xor   r15, 0x88        xor   r14, 0
add   r15, 0x800             nov   r15, r10        mov    r15, 0x6f8        add   rsi, 0x78
add   r15, 0x800             nov   r15, r10        mov    r10b, 0x6f8       add   r12, 0
add   r15, 0x800             nov   r15, r10        mov    r9, 0x12           and   rax, 0x20
add   r15, 0x800             nov   r15, r10        or    r15, 0x78         mov    r11, 0
add   r15, 0x800             nov   r15, r10        add   r14, 0x29        add   r13, 0
add   r15, 0x800             nov   r15, r10        xor   r14, 0           or    rbx, 0
add   r15, 0x800             nov   r15, r10        add   r14, 0x3f        shl   rax, 3
add   r15, 0x800             nov   r15, r10        and   r14, r15        add   r8, rax
add   r15, 0x800             nov   r15, r10        or    r11, r13        or    r15, r15
add   r15, 0x800             nov   r15, r10        add   r14, r15        sub   r15, 0x10
add   r15, 0x800             nov   r15, r10        xor   r11, r13        or    r11, r13
add   r15, 0x800             nov   r15, r10        mov    r15, r13        mov    r10, rbp
add   r15, 0x800             nov   r15, r10        add   r14, r15        mov    r13, rbp
add   r15, 0x800             nov   r15, r10        xor   r11, r13        mov    r10, rbp
add   r15, 0x800             nov   r15, r10        add   r14, r15        sub   r13, 0x80
add   r15, 0x800             nov   r15, r10        xor   r11, r13        add   rdx, 0xc0
add   r15, 0x800             nov   r15, r10        add   r14, r15        jmp   ebx
add   r15, 0x800             nov   r15, r10        xor   r11, r13        mov    rax, rbp
add   r15, 0x800             nov   r15, r10        add   r14, r15        and   rax, 0x20
add   r15, 0x800             nov   r15, r10        xor   r11, r13        mov    r10, 0
add   r15, 0x800             nov   r15, r10        add   r14, r15        add   r13, 0
add   r15, 0x800             nov   r15, r10        xor   r11, r13        or    rbx, 0
add   r15, 0x800             nov   r15, r10        add   r14, r15        shl   rax, 3
add   r15, 0x800             nov   r15, r10        xor   r11, r13        add   r8, rax
add   r15, 0x800             nov   r15, r10        add   r14, r15        or    r15, r15
add   r15, 0x800             nov   r15, r10        xor   r11, r13        sub   r15, 0x10
add   r15, 0x800             nov   r15, r10        add   r14, r15        or    r11, r13
add   r15, 0x800             nov   r15, r10        xor   r11, r13        mov    r15, rbp
add   r15, 0x800             nov   r15, r10        add   r14, r15        mov    r10, rbp
add   r15, 0x800             nov   r15, r10        xor   r11, r13        sub   r13, 0x80
add   r15, 0x800             nov   r15, r10        add   r14, r15        add   rdx, 0xc0
add   r15, 0x800             nov   r15, r10        xor   r11, r13        jmp   ebx

```

#3: No Central VM Dispatcher

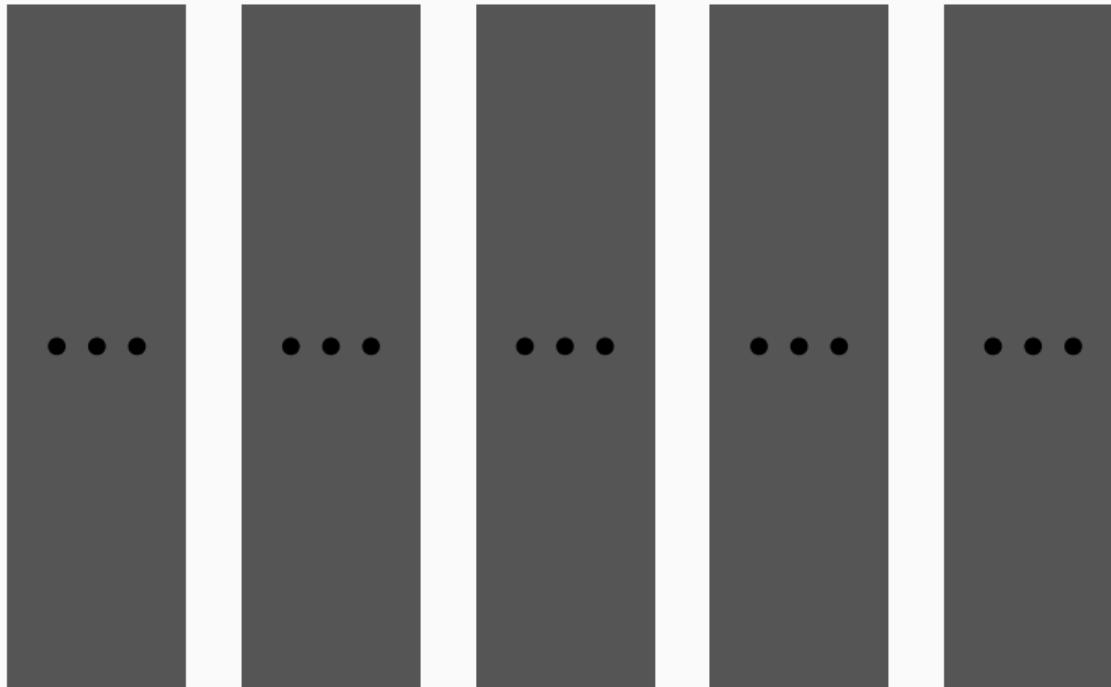
```

mov r15, 0x200          mov r15, rdx          add r8, 1           or r14, r14          mov r14, 0x200          add r15, 0x3f
xor r15, 0x800          xor r10d, dword ptr [r12]    or r8, 0x78        mov rax, rbp          or r15, 0x3f
mov rbx, rbp            sub r15, 0x800        add word ptr [rbx], r10w   and rcx, r13        and rdx, 0xc0
add rbx, 0xc0            or rdx, 0x400       mov r15, rax          add rax, 4           and r11, r14
mov rbi, qword ptr [rbx] nov rsi, 0x200      sub r15, rax          or r15, 0x88        and rsi, r9
mov r13, 1               nov r15, rbp         pop r9             mov r15, 0xffffffff  and rdx, 0xc0
mov rcx, 0               sub rsi, rsi        mov rcx, rbp        and rdx, 0xa         or r15, 0xc0
mov r15, rbp            sub rdi, rbp        cmp r8b, 0           mov r11, 0x78        and rdi, r14
add r15, 0xc0            or r8, 0x400       je 0x49e         mov rax, qword ptr [rdx]
or rcx, 0x88             sub rsi, r9         mov rdx, rbp        or r11, 0x40        and rsi, 0x4
add rbx, 0xb8            add rsi, rsi        add r15, 0x1         xor r11, 0x10        sub rsi, 4
mov r15, qword ptr [r15] add r14, 0           xor r11, 0x10       add rdx, 0xc0        or rbi, rsi
or r12, 0xffffffff80000000 add rsi, rax        add r15, 1           or r15, 0x800       movzx r9, word ptr [rax]
sub rcx, 0x78            and r8, 0x88       sub r15, 0x12       and r11, 0x20        and rdi, 0x7fffffff
movzx r10, word ptr [rbx] xor r12, r14       add rdx, 0xc0        xor r15, 0xc0        or r15, 0x20
xor r12, r13             nov rsi, rbp        xor r14, 4           mov r15, 0x12        and rsi, 0x58
add r12, 0xfffff         add rdi, 0xc0       add r8, 0x12         mov r15, 0x12        or r10, 0x20
add r15, 0                sub r8, rdi        mov rdx, qword ptr [rdx]
mov r8, rbp              add r8, 0x78       sub r11, r8           xor r10, 0x40        and r10, 0x40
sub rcx, 0x10             add rsi, 4         add rdx, 4           xor r10, 0x40        add r15, 0x3f505c07
or r12, r12               nov rcx, 0x200      or r15, 0x800       add r15, 0x88        add r15, 0x88
or rcx, 0x800             nov rdi, qword ptr [rdi]  mov r15, 0x800       mov r12, rbp
movzx r11, word ptr [r15] add dword ptr [rsi], 0x2549  mov r14, 0x8          or rdi, 0x98        or r12, 0x80
xor rcx, 0x800            xor r15, 0xd2       add rsi, 0x78        and r12, 0x8         add r12, 0x8
add r8, 0                 xor r15, 0x1         mov r10, 0x68        or rbx, 0x80        and r12, 0x8
add r11, 0                xor r15, 0x12       add r10, 0x68        and rax, 0xfffff
and r12, 0x29             add r15, 0x29       xor r10, 0x68        mov r11, 0x1         mov r13, 0x400
add rbx, 0x800            sub r15, 0x80       add r10, 0x68        and rsi, r8         add rsi, r8
nov r11, qword ptr [r11]  mov r15, rdi       xor r10, 0x68        or r10, 0x8         and r10, 0x8
add rbx, 1                add r15, rdi       add r10, 0x68        or rbx, 0x20        and rax, 0xfffff
and r12, r9               or r15, r12       xor r10, 0x68        and r11, r13        mov r11, 0x1
nov rdx, 1                or r15, r12       add r10, 0x68        or r15, 0x10        or r15, 0x10
xor r12, r9               xor r15, r12       xor r10, 0x68        and r11, r13        or r11, r13
add r11, r1                xor r15, r12       add r10, 0x68        or r15, 0x10        and r15, 0x10
and r12, r9               add r15, r12       xor r10, 0x68        and r11, r13        or r11, r13
nov rdx, 1                xor r15, r12       add r10, 0x68        or r15, 0x10        and r15, 0x10
sub r9, r11               add r15, r12       xor r10, 0x68        and r11, r13        or r11, r13
pushfq                  xor r15, r12       add r10, 0x68        or r15, 0x10        and r15, 0x10
xor rbx, 0xf0             add r15, r12       xor r10, 0x68        and r11, r13        or r11, r13
xor rbx, 0x800            xor r15, r12       add r10, 0x68        or r15, 0x10        and r15, 0x10
and rdx, r8               xor r15, r12       xor r10, 0x68        and r11, r13        or r11, r13
nov r12, rbp              xor r15, r12       add r10, 0x68        or r15, 0x10        and r15, 0x10
xor rdx, 0x20              xor r15, r12       xor r10, 0x68        and r11, r13        or r11, r13
sub rbx, 4                 xor r15, r12       add r10, 0x68        or r15, 0x10        and r15, 0x10
add r11, 0x2549b044       xor r15, r12       xor r10, 0x68        and r11, r13        or r11, r13
or rbx, 0x78               xor r15, r12       add r10, 0x68        or r15, 0x10        and r15, 0x10
and rdx, r10               xor r15, r12       xor r10, 0x68        and r11, r13        or r11, r13
nov rax, 0                 xor r15, r12       add r10, 0x68        or r15, 0x10        and r15, 0x10
add r12, 0x42               xor r15, r12       xor r10, 0x68        and r11, r13        or r11, r13
add rbx, 0x5a               xor r15, r12       add r10, 0x68        or r15, 0x10        and r15, 0x10

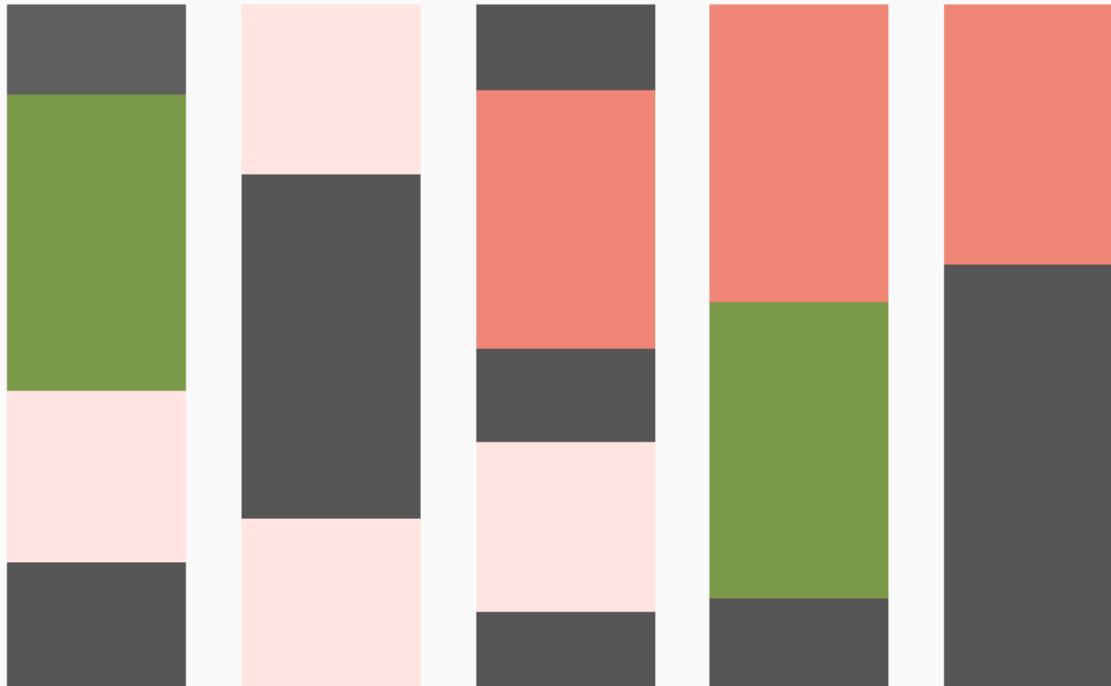
```

Split at indirect control-flow transfers

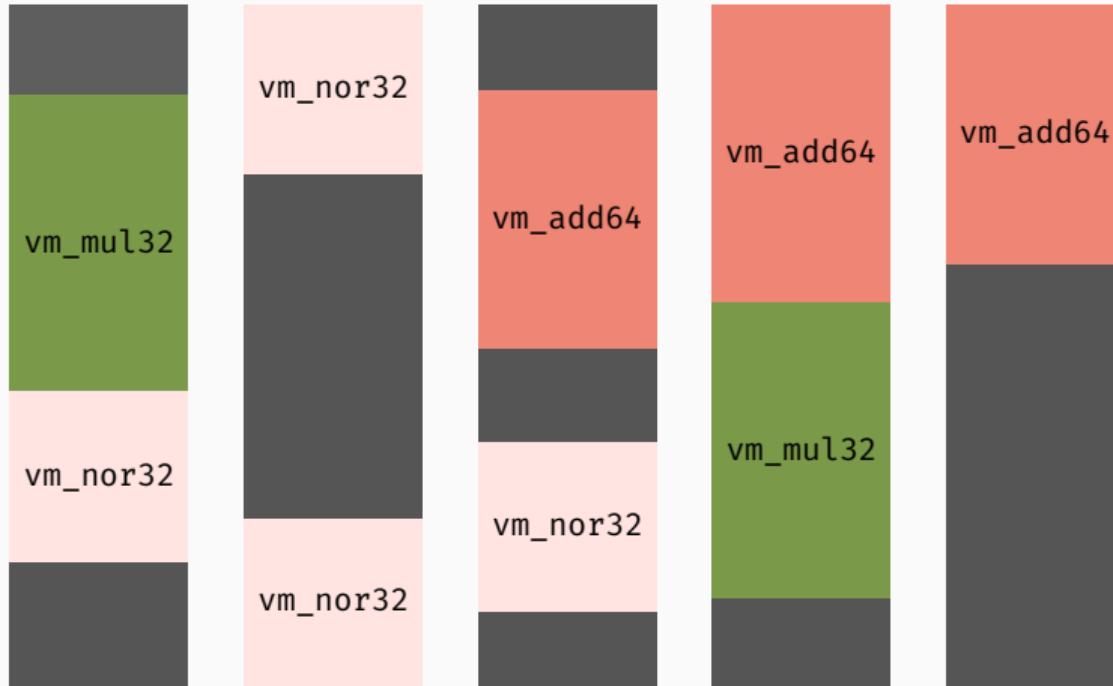
#4: No Explicit Handler Table



#4: No Explicit Handler Table



#4: No Explicit Handler Table



Conclusion

Take Aways

1. syntactic complexity insignificant

Take Aways

1. syntactic complexity insignificant
2. semantic complexity low within specified boundaries

Take Aways

1. syntactic complexity insignificant
2. semantic complexity low within specified boundaries
3. learn underlying code's semantics despite obfuscation

Take Aways

1. syntactic complexity insignificant
2. semantic complexity low within specified boundaries
3. learn underlying code's semantics despite obfuscation

Program Synthesis as an orthogonal approach to traditional techniques

Limitations

Implementation Shortcomings

choosing *meaningful* code window boundaries

$$(x \oplus y) + 2 \cdot (x \wedge y) \quad \text{vs.} \quad (x \oplus y) + 2$$

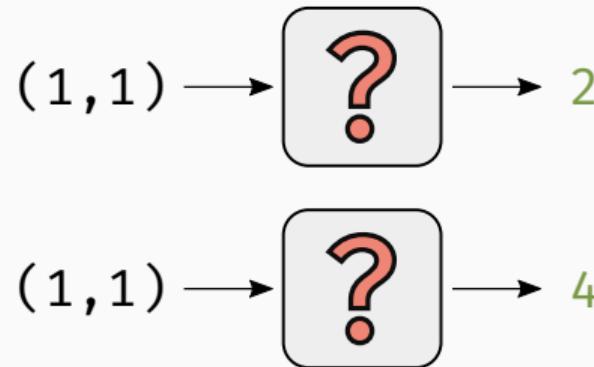
constants

$$x + 15324326921$$

control-flow operations

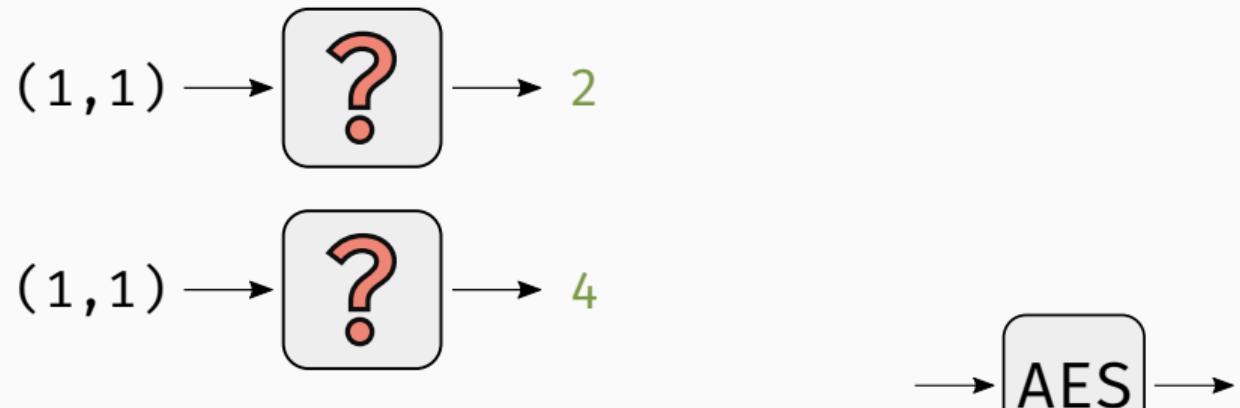
$$x \ ? \ y \ : \ z$$

Limitations



non-determinism

Limitations

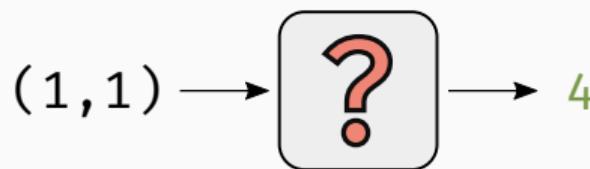
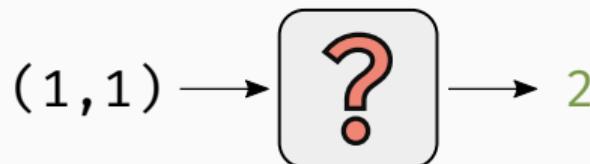


non-determinism

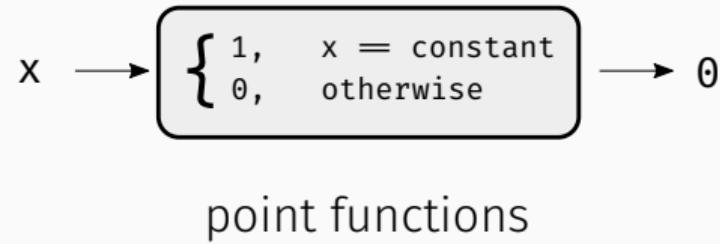


semantic complexity

Limitations



non-determinism



point functions



semantic complexity

Do try it at home!

Code Issues 1 Pull requests 0 Projects 0 Insights

Branch: master **syntia / samples /** Create new file Find file History

 mrphrazer	added MBA samples from tigress	Latest commit 91a5c16 7 days ago
..		
 info	added VM handler samples for vmprotect and themida	7 days ago
 mba/tigress	added MBA samples from tigress	7 days ago
 themida/tiger_white	added VM handler samples for vmprotect and themida	7 days ago
 vmprotect	added VM handler samples for vmprotect and themida	7 days ago
 tigress_mba_trace.bin	initial commit	15 days ago
 vmprotect_add16_trace.bin	initial commit	15 days ago

Summary

- obfuscation techniques (opaque predicates, VM, MBA)
- symbolic execution for syntactic deobfuscation
- program synthesis for semantic deobfuscation

<https://github.com/RUB-SysSec/syntia>