A General Approach to Underapproximate Reasoning about Concurrent Programs

4th International KLEE Workshop on Symbolic Execution @ ICSE 2024
15 April 2024

Azalea Raad
Imperial College

Julien Vanegue
Bloomberg

Josh Berdine
Meta

Peter O’Hearn
Lacework

Bloomberg

TechAtBloomberg.com

State of the Art: ***Correctness***

- Lots of work on *reasoning* for proving *correctness*
  - → Prove the *absence of bugs*
  - → *Overapproximate* reasoning
  - → ***Compositionality***

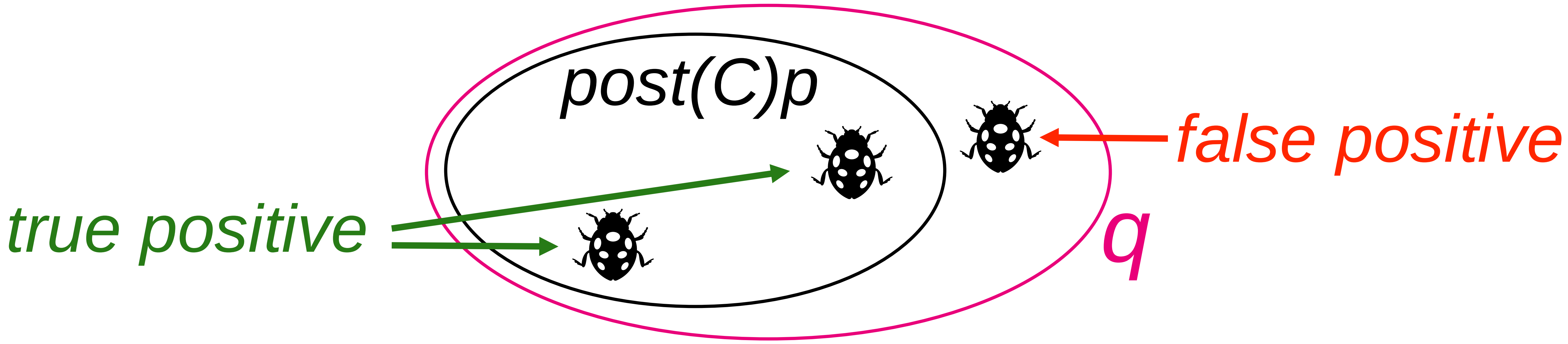
    in *code* ⇒ reasoning about *incomplete components*
    in *resources* accessed ⇒ spatial locality
  - → ***Scalability*** *to large teams and codebases*

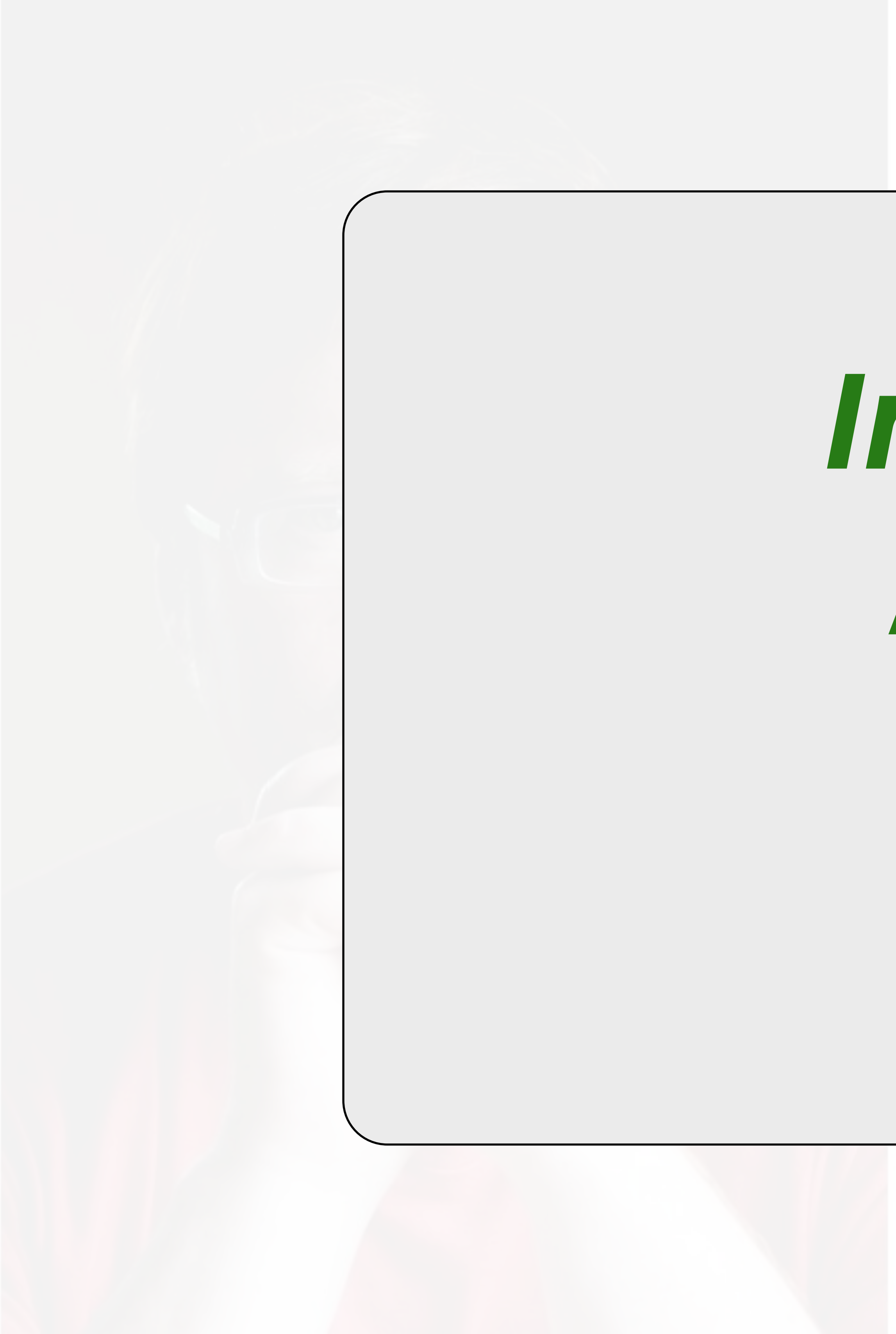
Hoare Logic (HL)

Hoare triples {p} C {q} *iff* post(C)p ⊆ q

q overapproximates post(C)p

For all states s in p
if running C on s terminates in s', then s' is in q





Incorrectness Logic:
A Formal Foundation for Bug Catching

Incorrectness Logic (IL)

Hoare triples $\{p\}\ C\ \{q\}$ *iff* $post(C)p \subseteq q$

For all states s in p
if running C on s terminates in s', then s' is in q

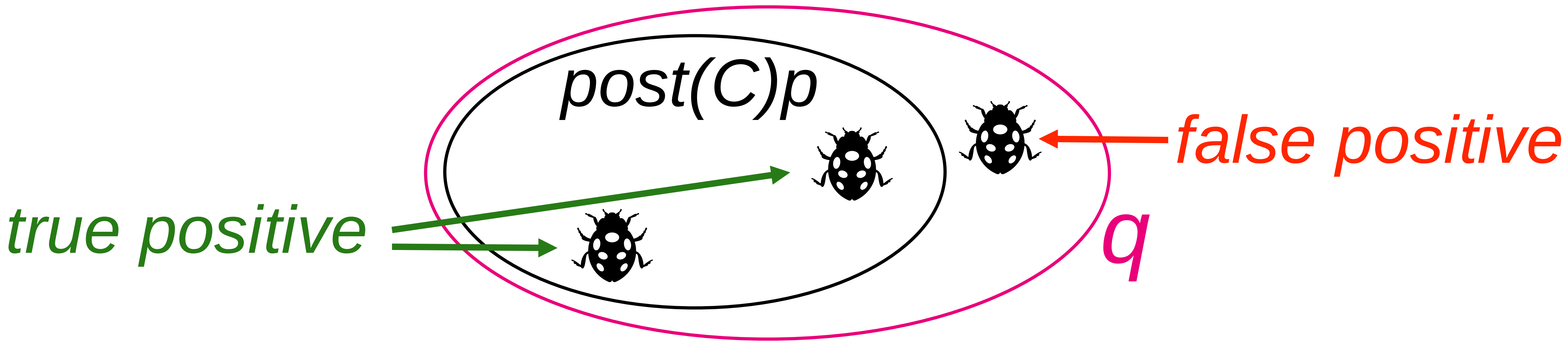
Incorrectness triples $[p]\ C\ [q]$ *iff* $post(C)p \supseteq q$

For all states s in q
s can be reached by running C on some s' in p

Incorrectness Logic (IL)

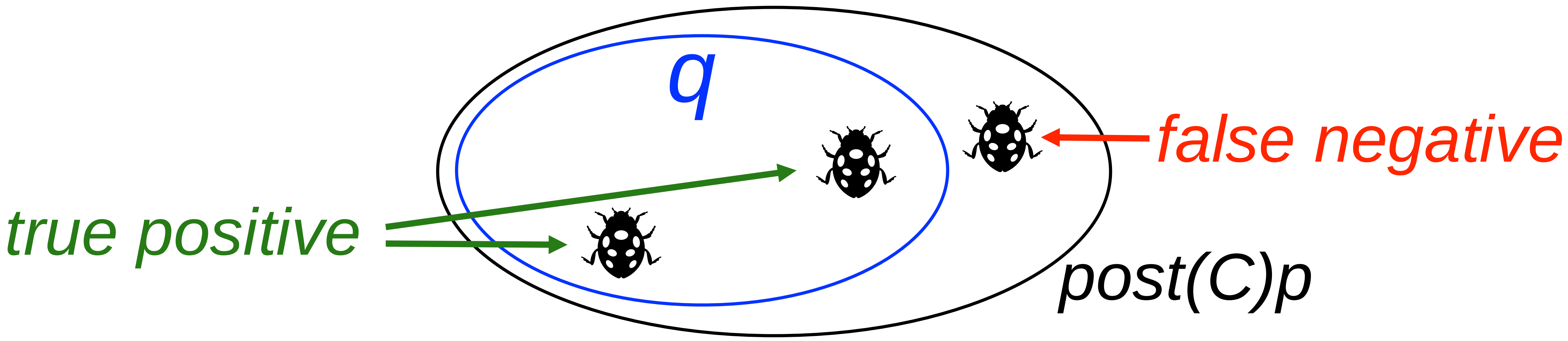
Hoare triples $\{p\}\ C\ \{q\}$ *iff* $post(C)p \subseteq q$

q overapproximates post(C)p



Incorrectness triples $[p]\ C\ [q]$ *iff* $post(C)p \supseteq q$

q underapproximates post(C)p



Incorrectness Logic (IL)

$$[p]\ C\ [\varepsilon: q]$$

ε: exit condition

ok: normal execution

er: erroneous execution

[y=v] x:=y [ok: x=y=v]

[p] error( ) [er: p]

IL Proof Rules and Principles (Sequencing)

$$\frac{[p]\ C_1\ [er: q]}{[p]\ C_1; C_2\ [er: q]}$$

$$\frac{[p]\ C_1\ [ok: r] \quad [r]\ C_2\ [\varepsilon: q]}{[p]\ C_1; C_2\ [\varepsilon: q]}$$

- Short-circuiting semantics for errors

IL Proof Rules and Principles (Branches)

$$\frac{[p]\ C_i\ [\varepsilon: q] \qquad \text{some } i \in \{1, 2\}}{[p]\ C_1 + C_2\ [\varepsilon: q]}$$

- Drop paths/branches (this is a sound underapproximation)
- Scalable bug detection!

Compositionality: Incorrectness Separation Logic (ISL)

IL

$[p]\ C\ [\varepsilon: q]$

SL

$$\frac{\{p\}\ C\ \{q\}}{\{p \ast r\}\ C\ \{q \ast r\}}$$

$x \mapsto - \ast x \mapsto - \Leftrightarrow \text{false}$

$x \mapsto v \ast \textbf{emp} \Leftrightarrow x \mapsto v$

ISL

$$\frac{[p]\ C\ [\varepsilon: q]}{[p \ast r]\ C\ [\varepsilon: q \ast r]}$$

Concurrency: Concurrent Incorrectness Separation Logic (CISL)

ISL

$$\frac{[p]\ C\ [\varepsilon: q]}{[p * r]\ C\ [\varepsilon: q * r]}$$

CSL

$$\frac{\{p_1\}\ C_1\ \{q_1\} \qquad \{p_2\}\ C_2\ \{q_2\}}{\{p_1 * p_2\}\ C_1 \parallel C_2\ \{q_1 * q_2\}}$$

CISL

$$\frac{[p_1]\ C_1\ [\varepsilon: q_1] \qquad [p_2]\ C_2\ [\varepsilon: q_2]}{[p_1 * p_2]\ C_1 \parallel C_2\ [\varepsilon: q_1 * q_2]}$$

Global Concurrency Bugs

Due to two or more threads, under certain interleavings:

1. *data-agnostic*: threads do not affect one another's control flow

$$\begin{array}{l} \texttt{free}(x); \\ [z] := 1; \end{array} \Bigg\| \begin{array}{l} a := [z]; \\ \texttt{if}\,(*)\ \text{L}: [x] := 1 \end{array}$$

data-agnostic use-after-free bug at L

CISL

2. *data-dependent* bugs: threads do affect one another's control flow

$$\begin{array}{l} \texttt{free}(x); \\ [z] := 1; \end{array} \Bigg\| \begin{array}{l} a := [z]; \\ \texttt{if}\,(a{=}1)\ \text{L}: [x] := 1 \end{array}$$

data-dependent use-after-free bug at L

not handled compositionally in CISL theory

Concurrent Adversarial Separation Logic (CASL)

CISL

$$\frac{[p_1]\ C_1\ [\varepsilon: q_1] \qquad [p_2]\ C_2\ [\varepsilon: q_2]}{[p_1 \ast p_2]\ C_1 \,||\, C_2\ [\varepsilon: q_1 \ast q_2]}$$

Rely-Guarantee

$$\frac{R_1, G_1 \vdash \{p\} C_1 \{q\} \qquad R_2, G_2 \vdash \{p\} C_2 \{q\}}{R_1 \cap R_2,\ G_1 \cup G_2 \vdash \{p\}\ C_1 \,||\, C_2\ \{q\}}$$

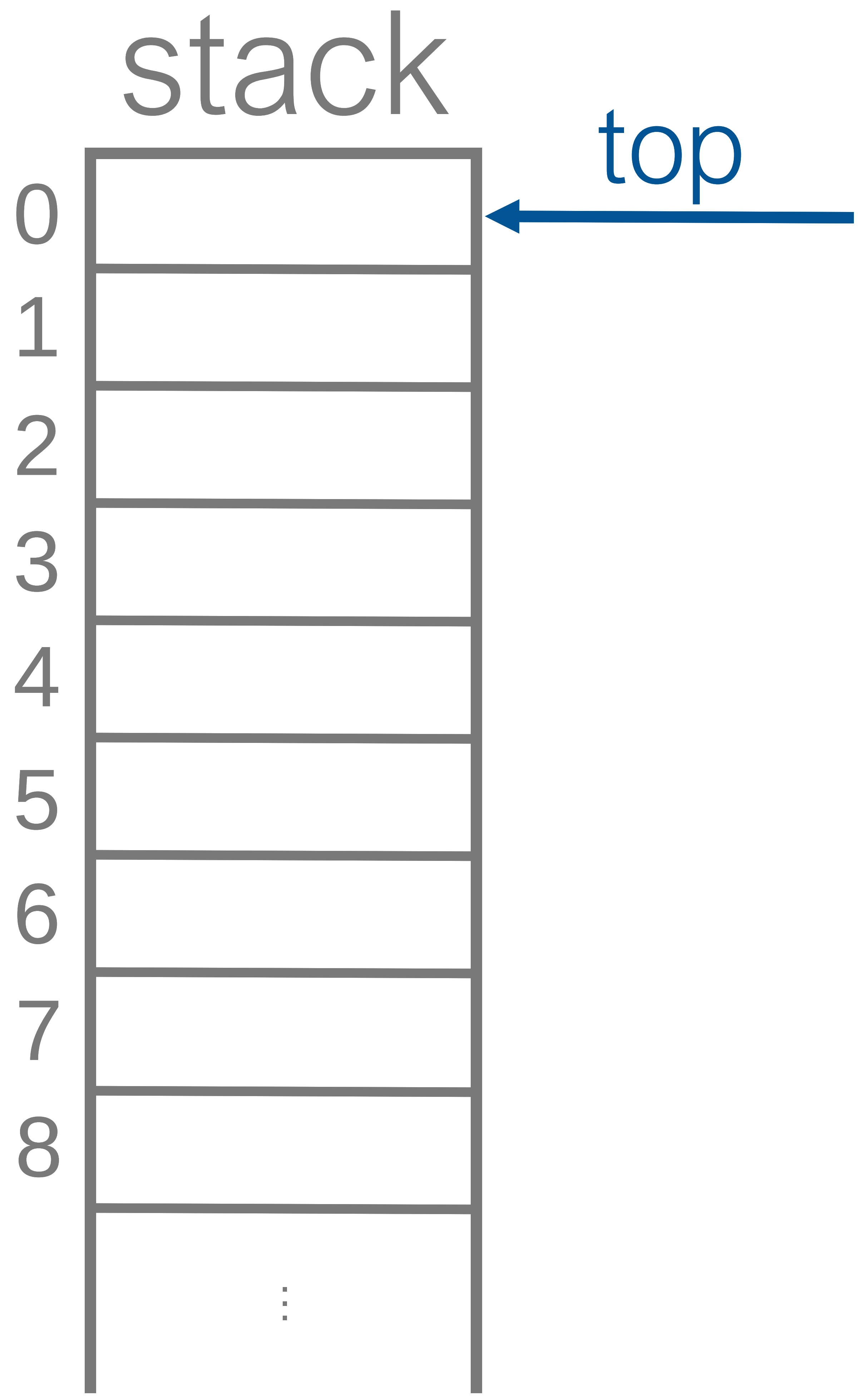
CASL

$$\frac{R_1, G_1, \boldsymbol{\theta} \vdash [p_1]\ C_1\ [\varepsilon: q_1] \qquad R_2, G_2, \boldsymbol{\theta} \vdash [p_2]\ C_2\ [\varepsilon: q_2]}{R_1 \cap R_2,\ G_1 \cup G_2,\ \boldsymbol{\theta} \vdash [p_1 \ast p_2]\ C_1 \,||\, C_2\ [\varepsilon: q_1 \ast q_2]}$$

CASL: Information Disclosure Attacks

$c \mapsto [\ ]$

```
send(c, 8);          ||   local sec := *;
recv(c, y);          ||   local w[8] := {0};
                     ||   recv(c, x);
                     ||   if (x ≤ 8)
                     ||     z := w[x];
                     ||     send(c, z);
```



CASL: Information Disclosure Attacks

$\text{send}(c, 8);$

$\text{recv}(c, y);$

$\|$

c ↦[ ]

$\text{local } sec := *;$

sec = top ✻ c ↦[]

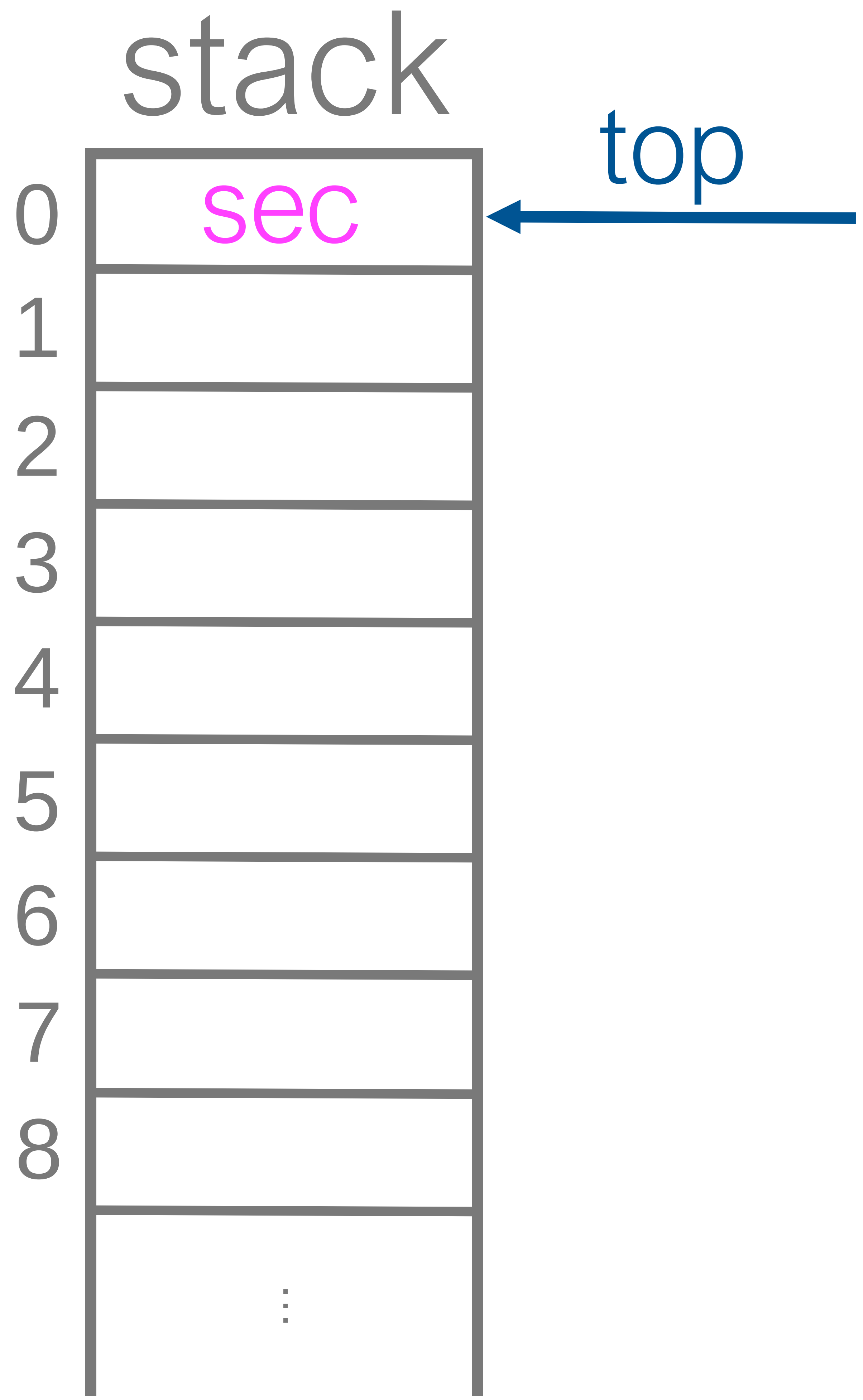
$\text{local } w[8] := \{0\};$

$\text{recv}(c, x);$

$\text{if } (x \leq 8)$

$\quad z := w[x];$

$\quad \text{send}(c, z);$



CASL: Information Disclosure Attacks

$c \mapsto [\,]$

send$(c, 8)$;
recv(c, y);
	$\Vert$	local $sec := *$;
		local $w[8] := \{0\}$;
		sec=top ✻ w[0]=top-8 ✻ $c \mapsto [\,]$
		recv(c, x);
		if $(x \leq 8)$
		  $z := w[x]$;
		  send(c, z);

CASL: Information Disclosure Attacks

$send(c, 8);$
$recv(c, y);$

||

c ↦[ ]

$\text{local } sec := *;$
$\text{local } w[8] := \{0\};$

sec=top ✻ w[8]=sec ✻c ↦[]

$recv(c, x);$
$\text{if } (x \leq 8)$
$\quad z := w[x];$
$\quad send(c, z);$

$G_a(\alpha_1)$: c ↦[ ] ⇝ c ↦[(8, τ_a, 0)]

$R_v = G_a$

CASL: Information Disclosure Attacks

c ↦[ ]

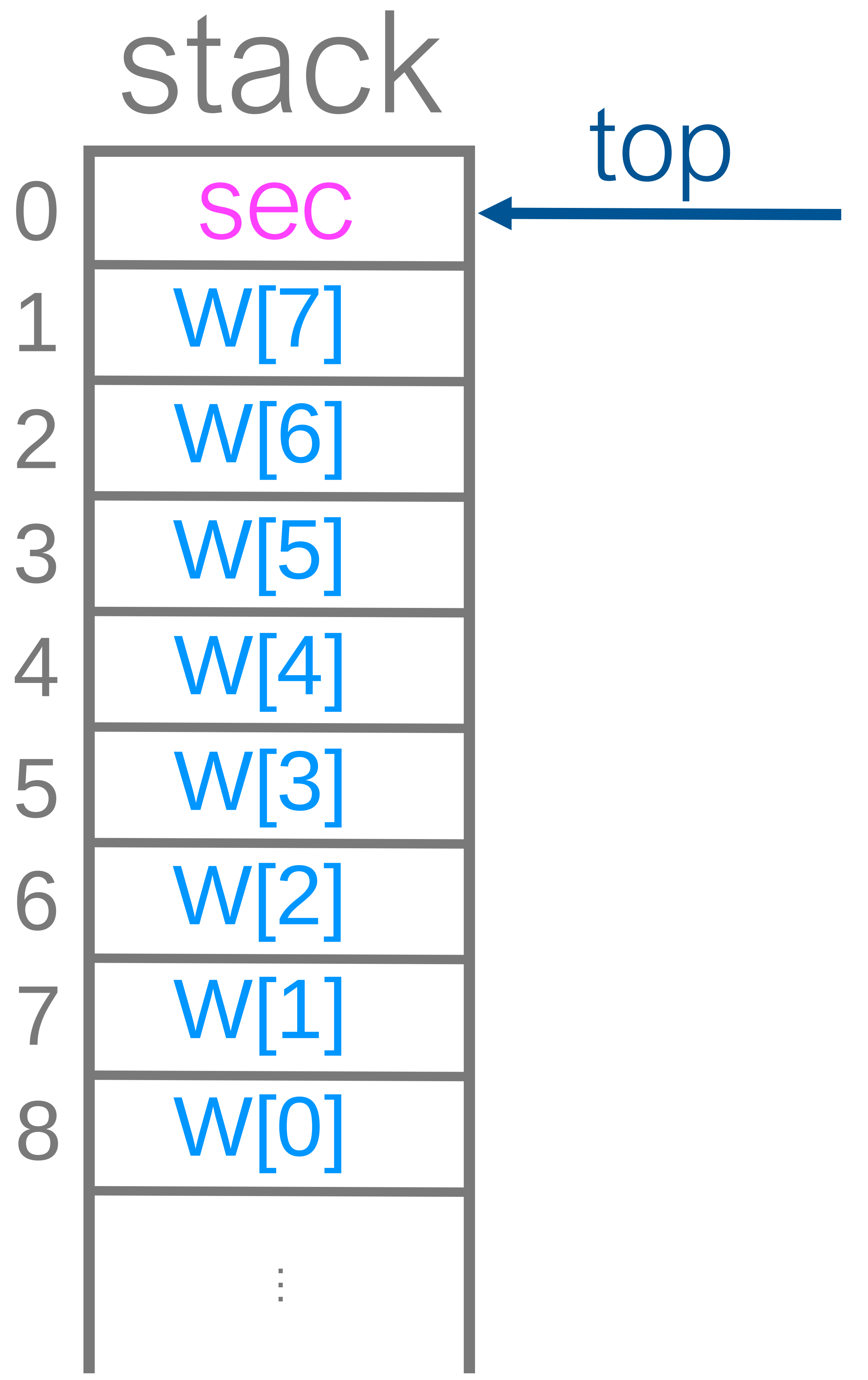
$\text{send}(c, 8);$
$\text{recv}(c, y);$

$\parallel$

$\text{local } sec := *;$
$\text{local } w[8] := \{0\};$ // $R_v(\alpha_1)$

sec=top ✻ w[8]=sec ✻c ↦[(8, τ_a, 0)]

$\text{recv}(c, x);$
$\text{if } (x \leq 8)$
$\quad z := w[x];$
$\quad \text{send}(c, z);$



$G_a(\alpha_1)$: c ↦[ ] ⇝ c ↦[(8, τ_a, 0)]

$R_v = G_a$

$G_v(\alpha_2)$: c ↦[(8, τ_a, 0)] ⇝ c ↦[ ]

CASL: Information Disclosure Attacks

c ↦[ ]

$\text{send}(c, 8);$

$\text{recv}(c, y);$

||

$\text{local } sec := *;$

$\text{local } w[8] := \{0\};$

$\text{recv}(c, x);$ // $G_v(\alpha_2)$

sec=top ✻ w[8]=sec ✻ x=(8,...) ✻c ↦[]

$\text{if } (x \leq 8)$

$z := w[x];$

$\text{send}(c, z);$

$G_a(\alpha_1)$: c ↦[ ] ⇝ c ↦[(8, τ_a, 0)]

$R_v = G_a$

$G_v(\alpha_2)$: c ↦[(8, τ_a, 0)] ⇝ c ↦[ ]

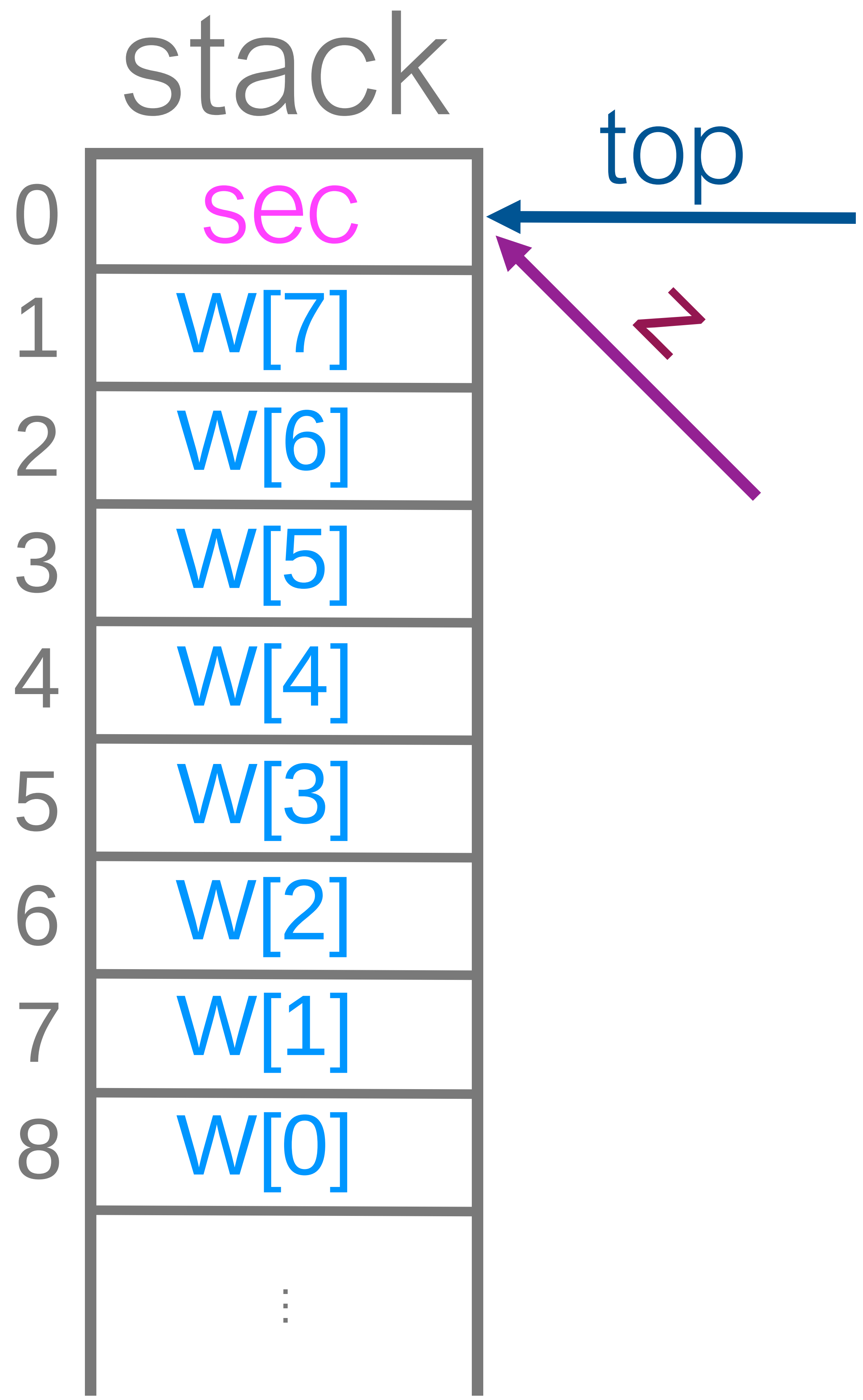
CASL: Information Disclosure Attacks

$c \mapsto [\,]$

$\text{send}(c, 8);$
$\text{recv}(c, y);$

$\|$

$\text{local } sec := *;$
$\text{local } w[8] := \{0\};$
$\text{recv}(c, x);$
$\text{if } (x \leq 8)$
$\quad z := w[x];$
$sec{=}top \circledast w[8]{=}sec \circledast z{=}sec \circledast c \mapsto [\,]$
$\quad \text{send}(c, z);$



$G_a(\alpha_1)$: $c \mapsto [\,] \rightsquigarrow c \mapsto [(8, \tau_a, 0)]$

$R_v = G_a$

$G_v(\alpha_2)$: $c \mapsto [\,(8, \tau_a, 0)] \rightsquigarrow c \mapsto [\,]$

$G_v(\alpha_3)$: $c \mapsto [\,] \rightsquigarrow c \mapsto [(-, \tau_v, 1)]$

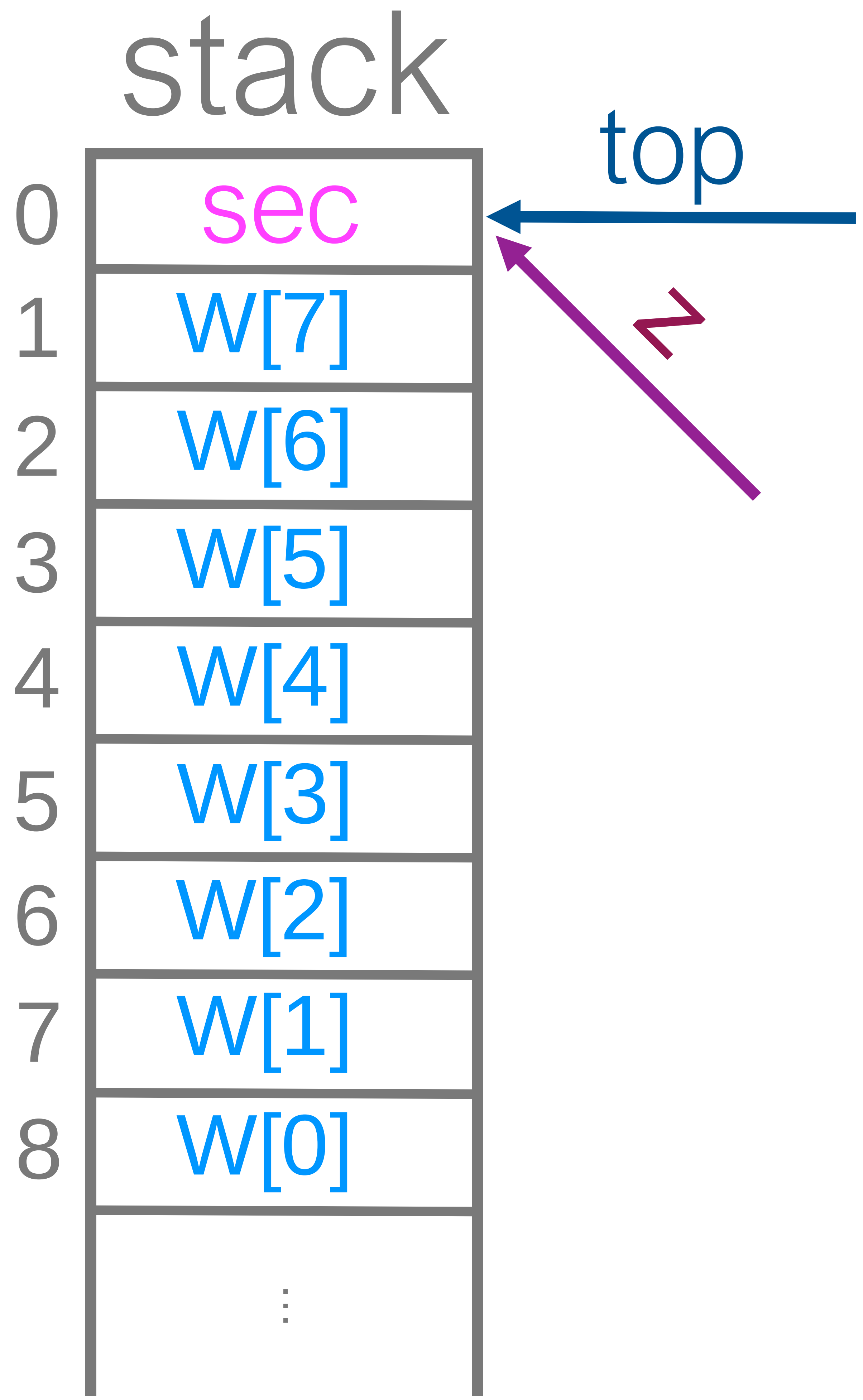
CASL: Information Disclosure Attacks

$c \mapsto [\ ]$

$\mathsf{send}(c, 8);$ $\mathsf{recv}(c, y);$	$\mathsf{local}\ sec := *;$ $\mathsf{local}\ w[8] := \{0\};$ $\mathsf{recv}(c, x);$ $\mathsf{if}\ (x \leq 8)$ $\quad z := w[x];$ $\quad \mathsf{send}(c, z);$ // $G_v(\alpha_3)$

z=sec, $c \mapsto [(sec, \tau_v, 1)]$ // $R_v(\alpha_4)$

error: information disclosure!



$G_a(\alpha_1)$: $c \mapsto [\ ] \rightsquigarrow c \mapsto [(8, \tau_a, 0)]$

$G_a(\alpha_4)$: $c \mapsto [(v, \tau_v, 1)] \rightsquigarrow c \mapsto [\ ]$

$R_v = G_a$

$G_v(\alpha_2)$: $c \mapsto [\ (8, \tau_a, 0)] \rightsquigarrow c \mapsto [\ ]$

$G_v(\alpha_3)$: $c \mapsto [\ ] \rightsquigarrow c \mapsto [(-, \tau_v, 1)]$

CASL: Information Disclosure Attacks

$c \mapsto [\,]$

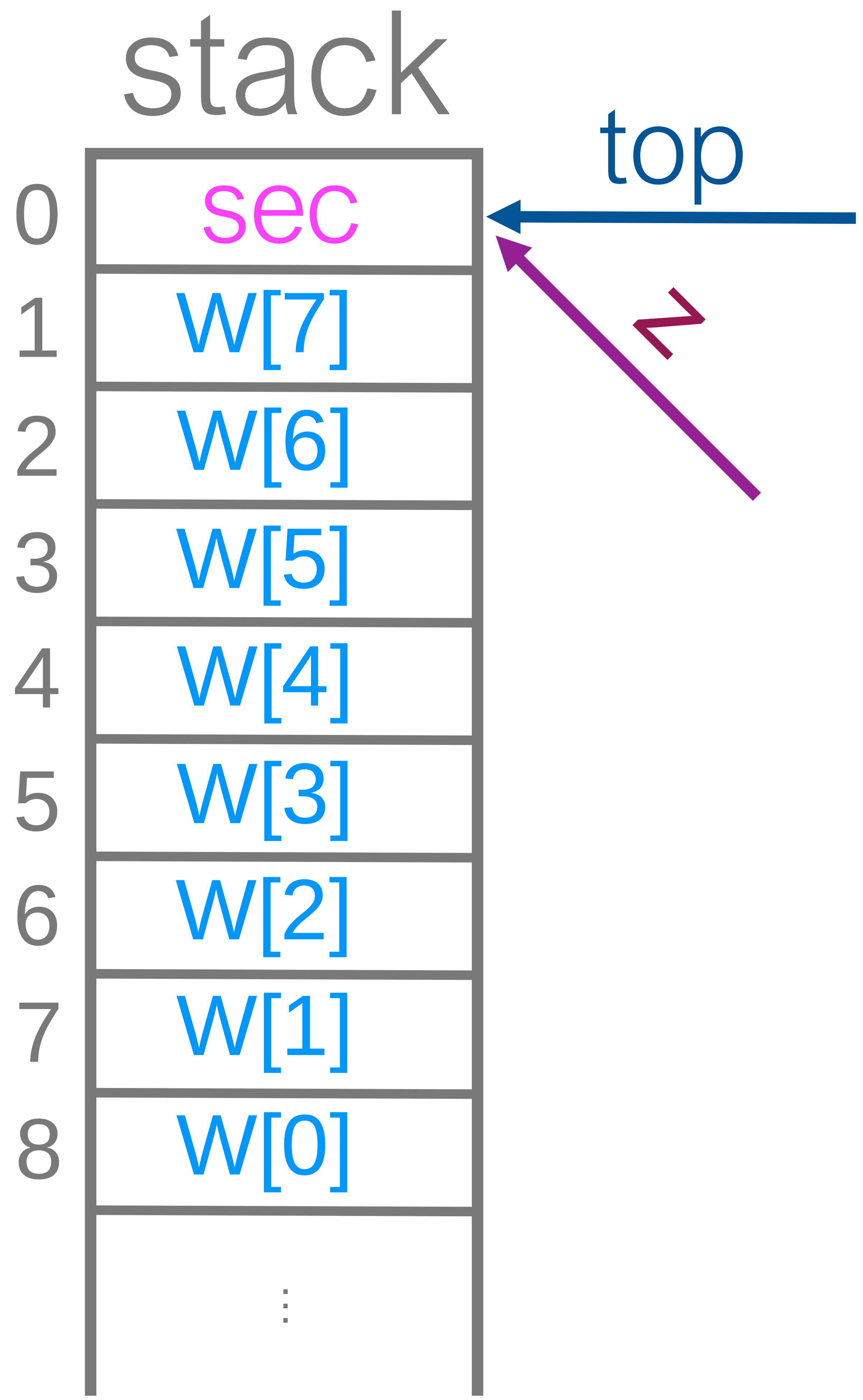
$$\mathsf{send}(c, 8);$$
$$\mathsf{recv}(c, y);$$

||

$c \mapsto [\,]$

$$
\begin{array}{l}
\mathsf{local}\ sec := *; \\
\mathsf{local}\ w[8] := \{0\}; \\
\mathsf{recv}(c, x); \\
\mathsf{if}\ (x \leq 8) \\
\quad z := w[x]; \\
\quad \mathsf{send}(c, z);
\end{array}
$$

$c \mapsto [(\mathrm{sec}, \tau_v, 1)]$



$G_a(\alpha_1)$: $c \mapsto [\,] \rightsquigarrow c \mapsto [(8, \tau_a, 0)]$

$R_v = G_a$

$G_a(\alpha_4)$: $c \mapsto [(v, \tau_v, 1)] \rightsquigarrow c \mapsto [\,]$

$G_v(\alpha_2)$: $c \mapsto [\,(8, \tau_a, 0)] \rightsquigarrow c \mapsto [\,]$

$R_a = G_v$

$G_v(\alpha_3)$: $c \mapsto [\,] \rightsquigarrow c \mapsto [(-, \tau_v, 1)]$

CASL: Information Disclosure Attacks

$c \mapsto [\,]$

$\mathsf{send}(c, 8);$ // $G_a(\alpha_1)$

$c \mapsto [(8, \tau_a, 0)]$

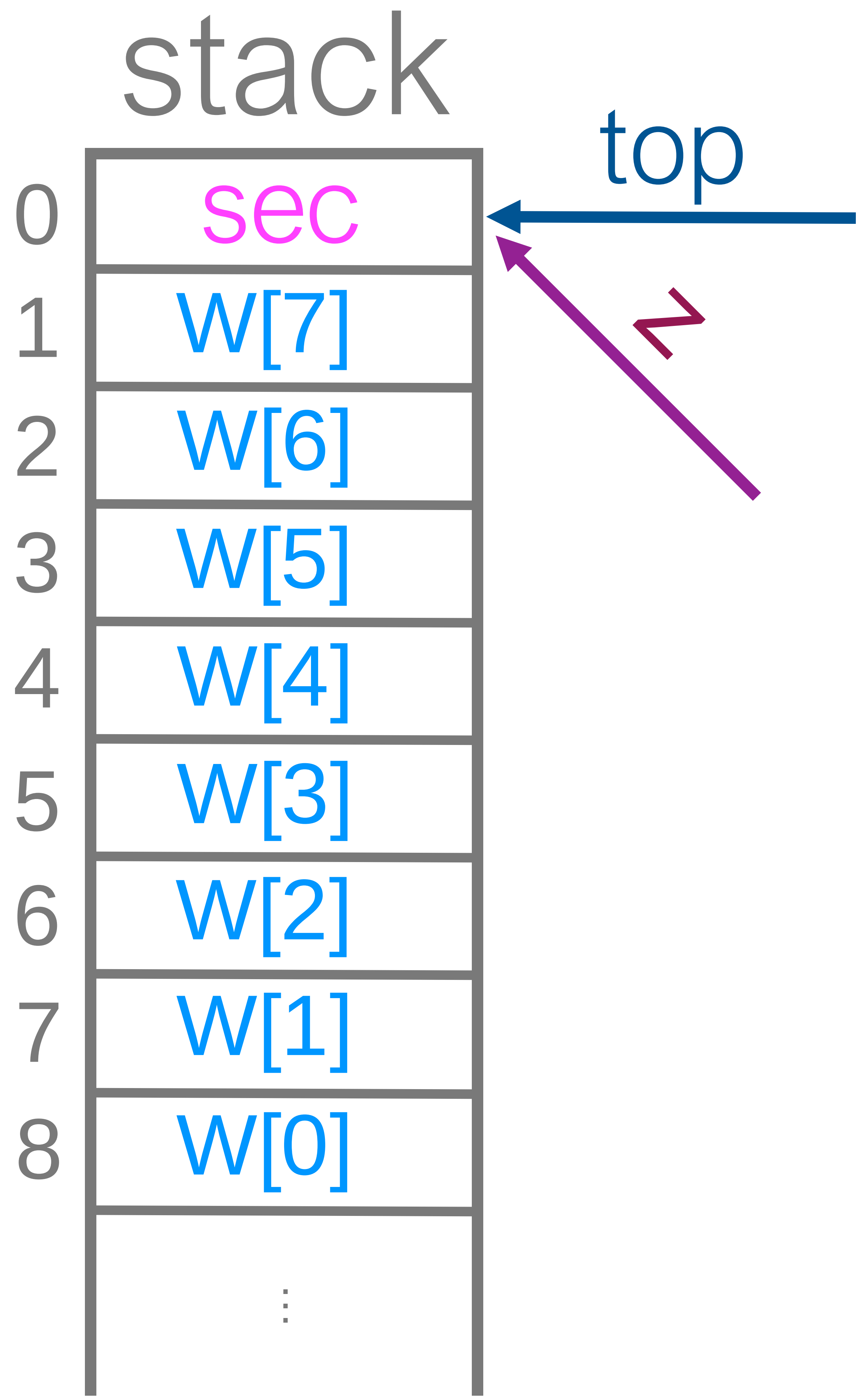
$\mathsf{recv}(c, y);$

||

$c \mapsto [\,]$

$\mathsf{local}\ sec := *;$
$\mathsf{local}\ w[8] := \{0\};$
$\mathsf{recv}(c, x);$
$\mathsf{if}\ (x \le 8)$
$\quad z := w[x];$
$\quad \mathsf{send}(c, z);$

$c \mapsto [(sec, \tau_v, 1)]$



$G_a(\alpha_1): c \mapsto [\,] \rightsquigarrow c \mapsto [(8, \tau_a, 0)]$

$G_a(\alpha_4): c \mapsto [(v, \tau_v, 1)] \rightsquigarrow c \mapsto [\,]$

$R_a = G_v$

$R_v = G_a$

$G_v(\alpha_2): c \mapsto [(8, \tau_a, 0)] \rightsquigarrow c \mapsto [\,]$

$G_v(\alpha_3): c \mapsto [\,] \rightsquigarrow c \mapsto [(-, \tau_v, 1)]$

CASL: Information Disclosure Attacks

$c \mapsto [\,]$

$\mathsf{send}(c, 8);$ // $G_a(\alpha_1)$

$c \mapsto [(8, \tau_a, 0)]$

// $R_a(\alpha_2)$;

$c \mapsto []$

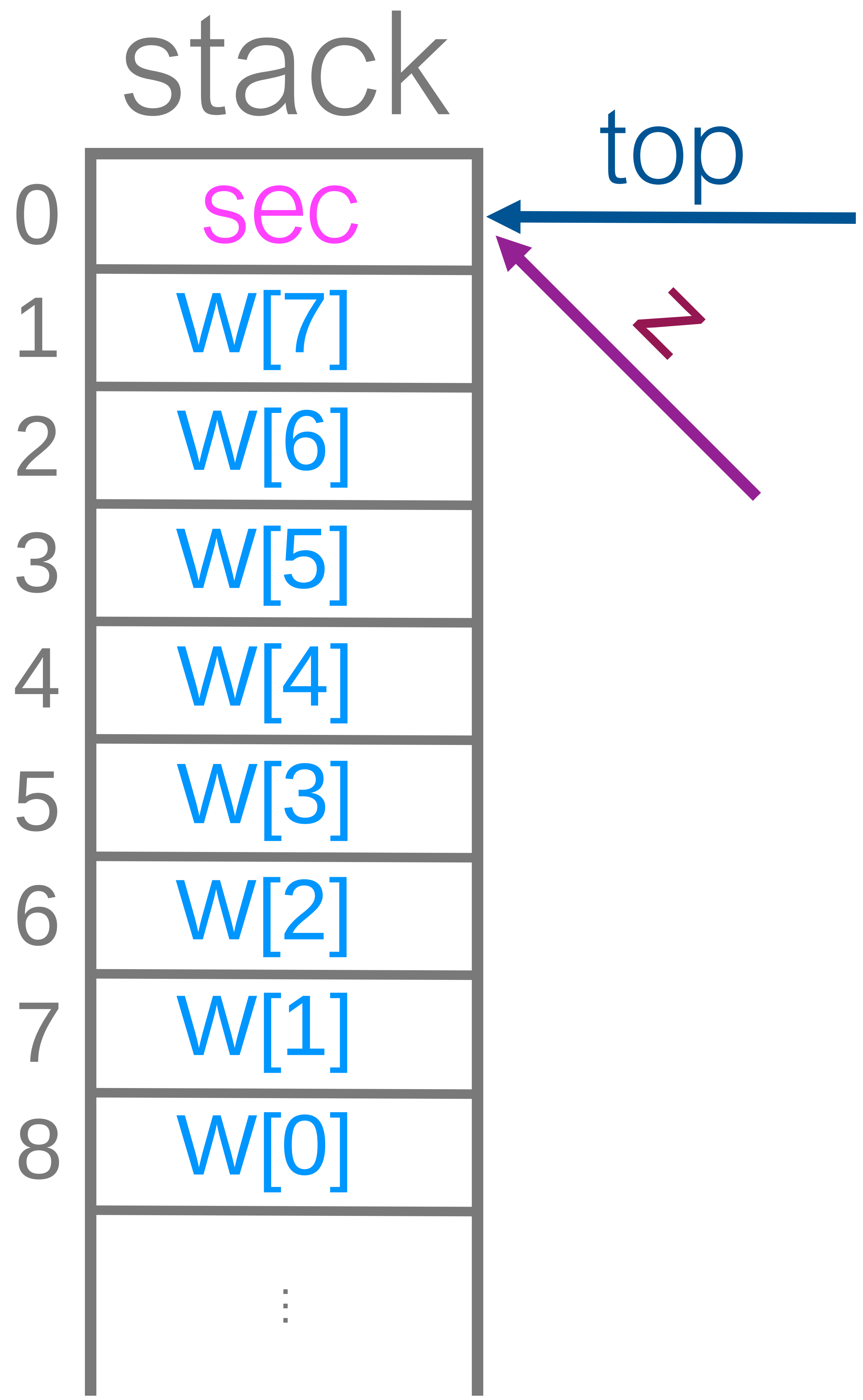
$\mathsf{recv}(c, y);$

$\|$

$c \mapsto [\,]$

local $sec := *$;
local $w[8] := \{0\}$;
$\mathsf{recv}(c, x)$;
if $(x \le 8)$
  $z := w[x]$;
  $\mathsf{send}(c, z)$;

$c \mapsto [(sec, \tau_v, 1)]$



$G_a(\alpha_1)$: $c \mapsto [\,] \rightsquigarrow c \mapsto [(8, \tau_a, 0)]$

$G_a(\alpha_4)$: $c \mapsto [(v, \tau_v, 1)] \rightsquigarrow c \mapsto [\,]$

$R_a = G_v$

$R_v = G_a$

$G_v(\alpha_2)$: $c \mapsto [\,(8, \tau_a, 0)] \rightsquigarrow c \mapsto [\,]$

$G_v(\alpha_3)$: $c \mapsto [\,] \rightsquigarrow c \mapsto [(-, \tau_v, 1)]$

CASL: Information Disclosure Attacks

$c \mapsto [\ ]$

$\mathrm{send}(c, 8);$ // $G_a(\alpha_1)$

$c \mapsto [(8, \tau_a, 0)]$

// $R_a(\alpha_2)$; $R_a(\alpha_3)$

$c \mapsto [(v, \tau_v, 1)]$

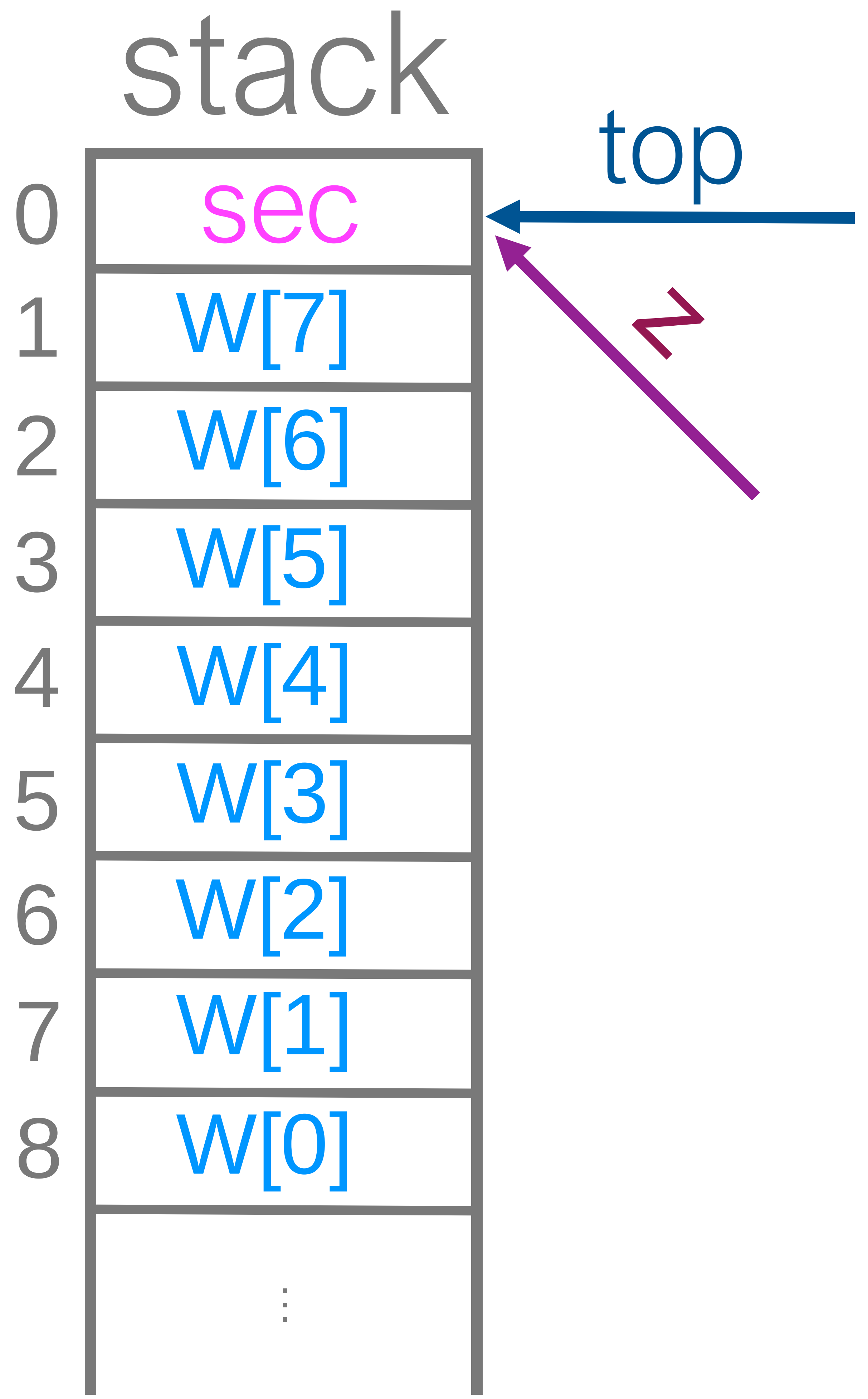
$\mathrm{recv}(c, y);$

||

$c \mapsto [\ ]$

local $sec := *;$
local $w[8] := \{0\};$
$\mathrm{recv}(c, x);$
if $(x \leq 8)$
  $z := w[x];$
  $\mathrm{send}(c, z);$

$c \mapsto [(sec, \tau_v, 1)]$



$G_a(\alpha_1)$: $c \mapsto [\ ] \rightsquigarrow c \mapsto [(8, \tau_a, 0)]$

$G_a(\alpha_4)$: $c \mapsto [(v, \tau_v, 1)] \rightsquigarrow c \mapsto [\ ]$

$R_a = G_v$

$R_v = G_a$

$G_v(\alpha_2)$: $c \mapsto [(8, \tau_a, 0)] \rightsquigarrow c \mapsto [\ ]$

$G_v(\alpha_3)$: $c \mapsto [\ ] \rightsquigarrow c \mapsto [(-, \tau_v, 1)]$

CASL: Information Disclosure Attacks

$c \mapsto [\,]$

$\mathsf{send}(c, 8);$ // $G_a(\alpha_1)$

// $R_a(\alpha_2)$; $R_a(\alpha_3)$

$c \mapsto [(v, \tau_v, 1)]$

$\mathsf{recv}(c, y);$ // $G_a(\alpha_4)$

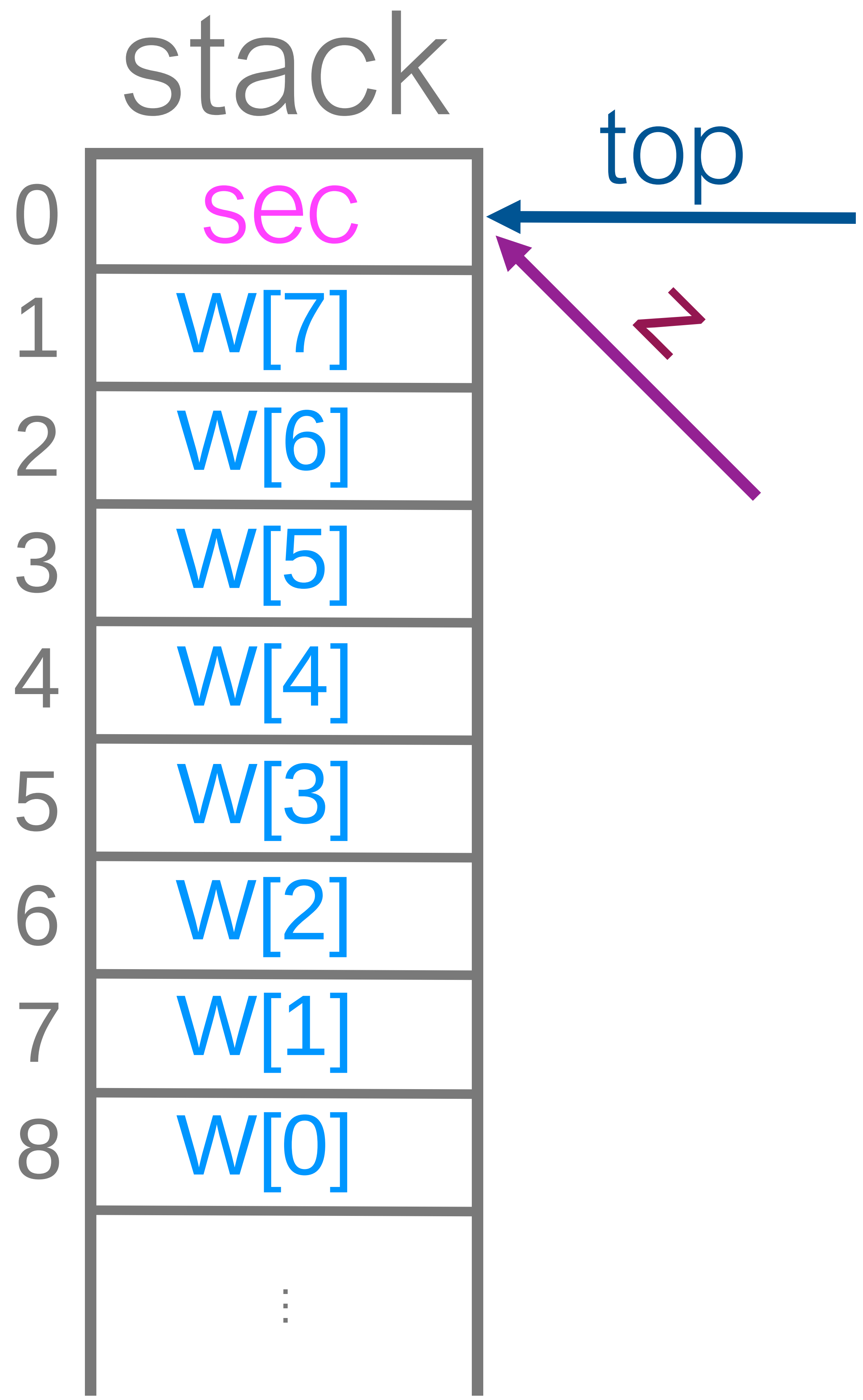
error: information disclosure!

$\|$

$c \mapsto [\,]$

```
local sec := *;
local w[8] := {0};
recv(c, x);
if (x ≤ 8)
  z := w[x];
  send(c, z);
```

$c \mapsto [(sec, \tau_v, 1)]$



$G_a(\alpha_1)$: $c \mapsto [\,] \rightsquigarrow c \mapsto [(8, \tau_a, 0)]$

$G_a(\alpha_4)$: $c \mapsto [(v, \tau_v, 1)] \rightsquigarrow c \mapsto [\,]$

$R_a = G_v$

$R_v = G_a$

$G_v(\alpha_2)$: $c \mapsto [(8, \tau_a, 0)] \rightsquigarrow c \mapsto [\,]$

$G_v(\alpha_3)$: $c \mapsto [\,] \rightsquigarrow c \mapsto [(-, \tau_v, 1)]$

Conclusions

- CASL: *A general* framework for *concurrent underapproximate* reasoning
- It subsumes CISL
- Can handle *both* data-agnostic and data-dependent bugs *compositionally*
- Used to detect *software vulnerabilities leading to exploits/attacks*
  - ☞ model a vulnerable program C_v and its adversary C_a as $C_a \parallel C_v$
- Instantiated for:
  - → Information disclosure attacks (over stacks & heaps)
  - → Buffer overflow attacks (over stacks & heaps)
  - → Memory safety attacks (e.g., zero allocation)

Thank You for Listening!

jvanegue@bloomberg.net

@jvanegue