

FSLH: Flexible Mechanized Speculative Load Hardening

MAX PLANCK INSTITUTE
FOR SECURITY AND PRIVACY



Jonathan Baumann^{1,2}, Roberto Blanco^{1,3}, Léon Ducruet^{1,4}, Sebastian Harwig^{1,5}, Cătălin Hrițcu¹

¹MPI-SP, Germany ²ENS Paris-Saclay, France ³TU/e, Netherlands ⁴ENS Lyon, France ⁵Ruhr University Bochum, Germany



- Spectre Attacks remain a threat



- Spectre Attacks remain a threat
- Existing mitigations have complementary strengths and weaknesses

- Spectre Attacks remain a threat
- Existing mitigations have complementary strengths and weaknesses

Selective SLH (Shivakumar et al. 2023)

- sparse protections:
low overhead

- Spectre Attacks remain a threat
- Existing mitigations have complementary strengths and weaknesses

Selective SLH (Shivakumar et al. 2023)

- sparse protections:
low overhead
- protects only cryptographic code

- Spectre Attacks remain a threat
- Existing mitigations have complementary strengths and weaknesses

Selective SLH (Shivakumar et al. 2023)

- sparse protections:
low overhead
- protects **only cryptographic** code

Ultimate SLH (Zhang et al. 2023)

- exhaustive protections:
high overhead

- Spectre Attacks remain a threat
- Existing mitigations have complementary strengths and weaknesses

Selective SLH (Shivakumar et al. 2023)

- sparse protections:
low overhead
- protects only cryptographic code

Ultimate SLH (Zhang et al. 2023)

- exhaustive protections:
high overhead
- protects all programs

- Spectre Attacks remain a threat
- Existing mitigations have complementary strengths and weaknesses

Selective SLH (Shivakumar et al. 2023)

- sparse protections:
low overhead
- protects only cryptographic code

Ultimate SLH (Zhang et al. 2023)

- exhaustive protections:
high overhead
- protects all programs

- Existing mitigations rely on manual security proofs

- Spectre Attacks remain a threat
- Existing mitigations have complementary strengths and weaknesses

Selective SLH (Shivakumar et al. 2023)

- sparse protections:
low overhead
- protects **only cryptographic** code

Ultimate SLH (Zhang et al. 2023)

- exhaustive protections:
high overhead
- protects **all** programs

- Existing mitigations rely on manual security proofs
 - First machine-checked proofs for Selective, Ultimate, *and* Flexible SLH

- Spectre Attacks remain a threat
- Existing mitigations have complementary strengths and weaknesses

Selective SLH (Shivakumar et al. 2023)

- sparse protections:
low overhead
- protects **only cryptographic** code

Ultimate SLH (Zhang et al. 2023)

- exhaustive protections:
high overhead
- protects **all** programs

- Existing mitigations rely on manual security proofs
 - First machine-checked proofs for Selective, Ultimate, *and* Flexible SLH
- Rocq development: ~ 4300 lines



```
if i < a1_size then  
  j <- a1[i];  
  x <- a2[j]
```

```
if i < a1_size then  
  j <- a1[i];  
  x <- a2[j]
```

Leaks j via the address



```
if i < a1_size then  
  j <- a1[i];  
  x <- if j == 0 then ...
```



Leaks j via control flow

Ensures *i* is in-bounds for a public array

```
if i < a1_size then  
  j <- a1[i];  
  x <- if j == 0 then ...
```

Leaks *j* via control flow

Ensures *i* is in-bounds for a public array

Bypassed speculatively

```
if i < a1_size then  
  j <- a1[i];  
  x <- if j == 0 then ...
```

Leaks *j* via control flow

Ensures *i* is in-bounds for a public array

Bypassed speculatively

```
if i < a1_size then  
  j <- a1[i];  
  x <- if j == 0 then ...
```

May speculatively read secret data

Leaks *j* via control flow


```
if i < a1_size then  
    j <- a1[i];  
    x <- a2[j]  
else
```

Mitigation introduced in LLVM



protected variable stores misspeculation flag

```
if i < a1_size then
  b := (i < a1_size) ? b : 1;
  j <- a1[i];
  x <- a2[j]
else
  b := (i < a1_size) ? 1 : b;
```

Mitigation introduced in LLVM

- keep track of misspeculation



protected variable stores misspeculation flag

updated using **constant-time conditional**

```
if i < a1_size then
  b := (i < a1_size) ? b : 1;
  j <- a1[i];
  x <- a2[j]
else
  b := (i < a1_size) ? 1 : b;
```

Mitigation introduced in LLVM

- keep track of misspeculation



protected variable stores misspeculation flag

updated using **constant-time conditional**

```
if i < a1_size then
  b := (i < a1_size) ? b : 1;
  j <- a1[i];
  x <- a2[j]
else
  b := (i < a1_size) ? 1 : b;
```

Mitigation introduced in LLVM

- keep track of misspeculation
- prevent speculative leaks



protected variable stores misspeculation flag

updated using **constant-time conditional**

```
if i < a1_size then
  b := (i < a1_size) ? b : 1;
  j <- a1[b == 1 ? 0 : i];
  x <- a2[b == 1 ? 0 : j]
else
  b := (i < a1_size) ? 1 : b;
```

Mitigation introduced in LLVM

- keep track of misspeculation
- prevent speculative leaks
 - **iSLH**: mask **indices** of loads

masks **index**

protected variable stores misspeculation flag

updated using **constant-time conditional**

```
if i < a1_size then
  b := (i < a1_size) ? b : 1;
  j <- b == 1 ? 0 : a[i];
  x <- b == 1 ? 0 : a[j]
else
  b := (i < a1_size) ? 1 : b;
```

masks **value**

Mitigation introduced in LLVM

- keep track of misspeculation
- prevent speculative leaks
 - **iSLH**: mask **indices** of loads
 - **vSLH**: mask loaded **values**

- only protects programs in the CCT discipline

- only protects programs in the CCT discipline

CCT discipline

- only protects programs in the CCT discipline

CCT discipline

- variables and arrays are statically labeled public or secret

- only protects programs in the **CCT discipline**

CCT discipline

- variables and arrays are statically labeled **public** or **secret**
- all branch conditions and indices must be **public**

- only protects programs in the CCT discipline

CCT discipline

- variables and arrays are statically labeled **public** or **secret**
- all branch conditions and indices must be **public**
- prevents *sequential* leakage of secrets

- only protects programs in the **CCT discipline**
- **selectively** masks loads into **public** variables

CCT discipline

- variables and arrays are statically labeled **public** or **secret**
- all branch conditions and indices must be **public**
- prevents *sequential* leakage of secrets



- only protects programs in the **CCT discipline**
- **selectively** masks loads into **public** variables
 - loads to secret variables are known not to leak

CCT discipline

- variables and arrays are statically labeled **public** or **secret**
- all branch conditions and indices must be **public**
- prevents *sequential* leakage of secrets

- only protects programs in the **CCT discipline**
- **selectively** masks loads into **public** variables
 - loads to secret variables are known not to leak
- enforces **SCT security**

CCT discipline

- variables and arrays are statically labeled **public** or **secret**
- all branch conditions and indices must be **public**
- prevents *sequential* leakage of secrets



- only protects programs in the **CCT discipline**
- **selectively** masks loads into **public** variables
 - loads to secret variables are known not to leak
- enforces **SCT security**

SCT security

$$s_1 \sim_P s_2 \wedge \langle c, s_1, \mathbb{f} \rangle \xrightarrow[\mathcal{D}]{\mathcal{O}_1}_S^* \cdot \\ \wedge \langle c, s_2, \mathbb{f} \rangle \xrightarrow[\mathcal{D}]{\mathcal{O}_2}_S^* \cdot \Rightarrow \mathcal{O}_1 = \mathcal{O}_2$$



- only protects programs in the CCT discipline
- **selectively** masks loads into **public** variables
 - loads to secret variables are known not to leak
- enforces **SCT security**

initial states agree on public variables

SCT security

$$s_1 \sim_P s_2 \wedge \langle c, s_1, \mathbb{f} \rangle \xrightarrow[\mathcal{D}]{\mathcal{O}_1}^*_S \cdot \wedge \langle c, s_2, \mathbb{f} \rangle \xrightarrow[\mathcal{D}]{\mathcal{O}_2}^*_S \cdot \Rightarrow \mathcal{O}_1 = \mathcal{O}_2$$

- only protects programs in the **CCT discipline**
- **selectively** masks loads into **public** variables
 - loads to secret variables are known not to leak
- enforces **SCT security**

initial states agree on public variables

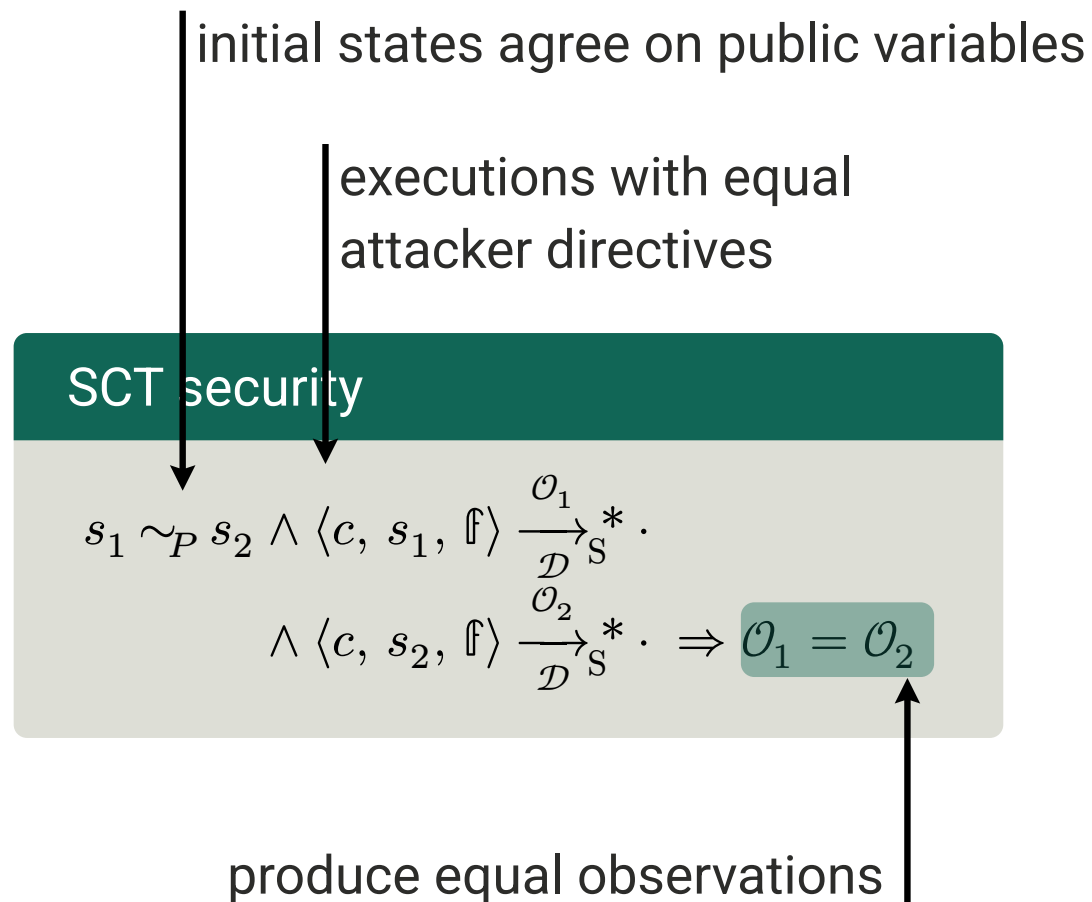
executions with equal
attacker directives

SCT security

$$s_1 \sim_P s_2 \wedge \langle c, s_1, \mathbb{f} \rangle \xrightarrow[\mathcal{D}]{\mathcal{O}_1}^*_S \cdot \wedge \langle c, s_2, \mathbb{f} \rangle \xrightarrow[\mathcal{D}]{\mathcal{O}_2}^*_S \cdot \Rightarrow \mathcal{O}_1 = \mathcal{O}_2$$



- only protects programs in the **CCT discipline**
- **selectively** masks loads into **public** variables
 - loads to secret variables are known not to leak
- enforces **SCT security**





- masks **all sources** of leakage,
always



- masks **all sources** of leakage,
always
 - loads, stores, branch conditions



- masks **all sources** of leakage,
always
 - loads, stores, branch conditions
 - also masks many other leaking instructions



- masks **all sources** of leakage,
always
 - loads, stores, branch conditions
 - also masks many other leaking instructions
- enforces **Relative Security**

- masks **all sources** of leakage, **always**
 - loads, stores, branch conditions
 - also masks many other leaking instructions
- enforces **Relative Security**

Relative Security

$$\begin{aligned} & \left(\forall \mathcal{O}_1 \mathcal{O}_2. \langle c, s_1 \rangle \xrightarrow{\mathcal{O}_1}^* . \right. \\ & \quad \left. \wedge \langle c, s_2 \rangle \xrightarrow{\mathcal{O}_2}^* . \Rightarrow \mathcal{O}_1 \preceq \mathcal{O}_2 \right) \\ \Rightarrow & \left(\forall \mathcal{O}_1 \mathcal{O}_2 \mathcal{D}. \langle \llbracket c \rrbracket, s_1, \mathbb{F} \rangle \xrightarrow[\mathcal{D}]{\mathcal{O}_1}^* s . \right. \\ & \quad \left. \wedge \langle \llbracket c \rrbracket, s_2, \mathbb{F} \rangle \xrightarrow[\mathcal{D}]{\mathcal{O}_2}^* s . \Rightarrow \mathcal{O}_1 = \mathcal{O}_2 \right) \end{aligned}$$

- masks **all sources** of leakage, **always**
 - loads, stores, branch conditions
 - also masks many other leaking instructions
- enforces **Relative Security**

Relative Security

$$\left(\begin{array}{l} \forall \mathcal{O}_1 \mathcal{O}_2. \langle c, s_1 \rangle \xrightarrow{\mathcal{O}_1}^* . \\ \wedge \langle c, s_2 \rangle \xrightarrow{\mathcal{O}_2}^* . \Rightarrow \mathcal{O}_1 \preceq \mathcal{O}_2 \end{array} \right)$$
$$\Rightarrow \left(\begin{array}{l} \forall \mathcal{O}_1 \mathcal{O}_2 \mathcal{D}. \langle \llbracket c \rrbracket, s_1, \mathbb{F} \rangle \xrightarrow[\mathcal{D}]{\mathcal{O}_1}^* s . \\ \wedge \langle \llbracket c \rrbracket, s_2, \mathbb{F} \rangle \xrightarrow[\mathcal{D}]{\mathcal{O}_2}^* s . \Rightarrow \mathcal{O}_1 = \mathcal{O}_2 \end{array} \right)$$

- masks **all sources** of leakage, **always**
 - loads, stores, branch conditions
 - also masks many other leaking instructions
- enforces **Relative Security**

Relative Security

$$\begin{aligned} & \left(\forall \mathcal{O}_1 \mathcal{O}_2. \langle c, s_1 \rangle \xrightarrow[\mathcal{D}]{\mathcal{O}_1}^* \cdot \right. \\ & \quad \left. \wedge \langle c, s_2 \rangle \xrightarrow[\mathcal{D}]{\mathcal{O}_2}^* \cdot \Rightarrow \mathcal{O}_1 \leq \mathcal{O}_2 \right) \\ \Rightarrow & \left(\forall \mathcal{O}_1 \mathcal{O}_2 \mathcal{D}. \langle \llbracket c \rrbracket, s_1, \mathbb{F} \rangle \xrightarrow[\mathcal{D}]{\mathcal{O}_1}^* \cdot \right. \\ & \quad \left. \wedge \langle \llbracket c \rrbracket, s_2, \mathbb{F} \rangle \xrightarrow[\mathcal{D}]{\mathcal{O}_2}^* \cdot \Rightarrow \mathcal{O}_1 = \mathcal{O}_2 \right) \end{aligned}$$

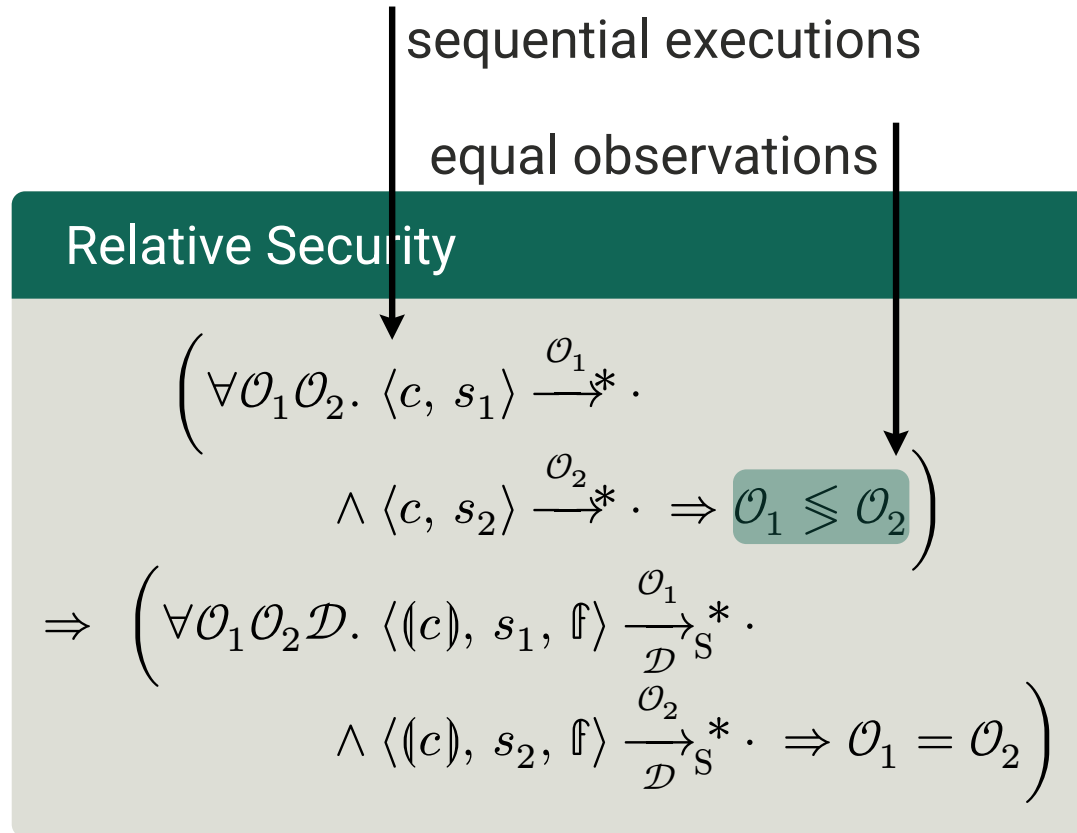
- masks **all sources** of leakage, **always**
 - loads, stores, branch conditions
 - also masks many other leaking instructions
- enforces **Relative Security**

sequential executions

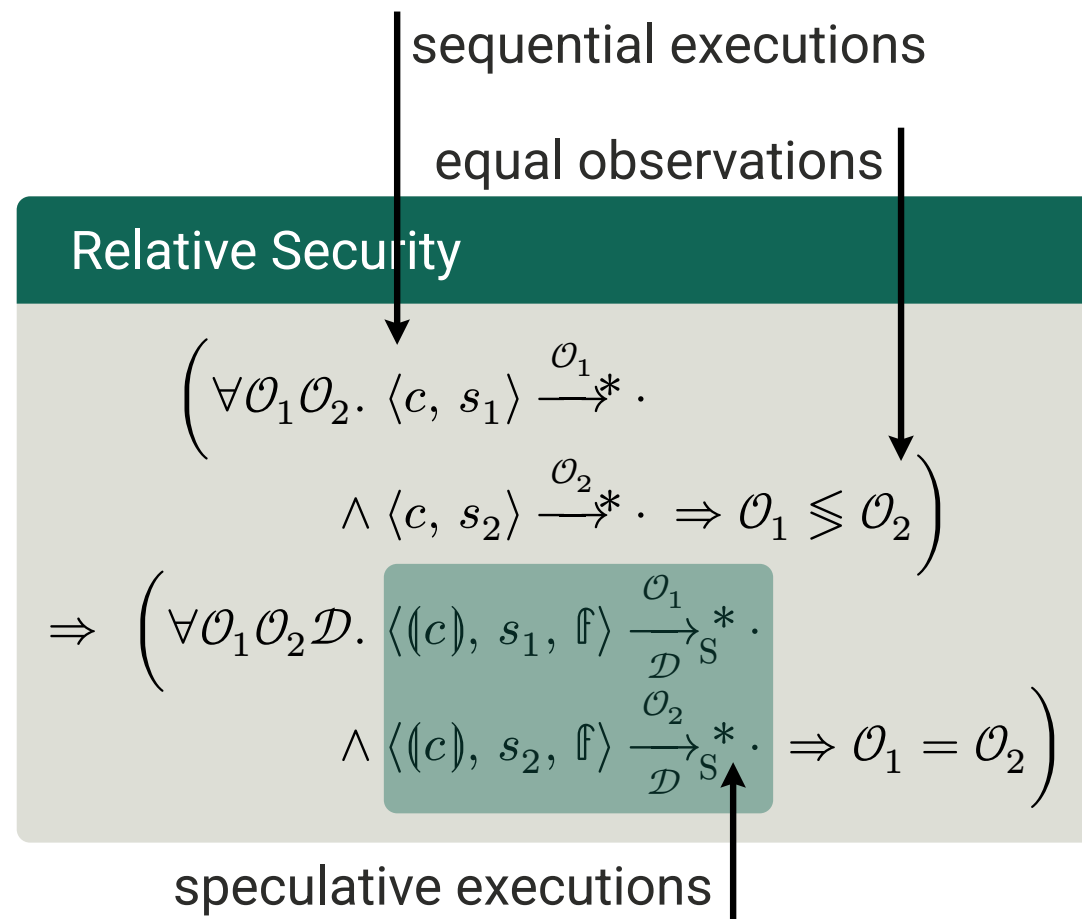
Relative Security

$$\begin{aligned}
 & \left(\forall \mathcal{O}_1 \mathcal{O}_2. \langle c, s_1 \rangle \xrightarrow{\mathcal{O}_1}^* \cdot \right. \\
 & \quad \left. \wedge \langle c, s_2 \rangle \xrightarrow{\mathcal{O}_2}^* \cdot \Rightarrow \mathcal{O}_1 \leq \mathcal{O}_2 \right) \\
 \Rightarrow & \left(\forall \mathcal{O}_1 \mathcal{O}_2 \mathcal{D}. \langle \llbracket c \rrbracket, s_1, \mathbb{F} \rangle \xrightarrow[\mathcal{D}]{\mathcal{O}_1}^* s^* \cdot \right. \\
 & \quad \left. \wedge \langle \llbracket c \rrbracket, s_2, \mathbb{F} \rangle \xrightarrow[\mathcal{D}]{\mathcal{O}_2}^* s^* \cdot \Rightarrow \mathcal{O}_1 = \mathcal{O}_2 \right)
 \end{aligned}$$

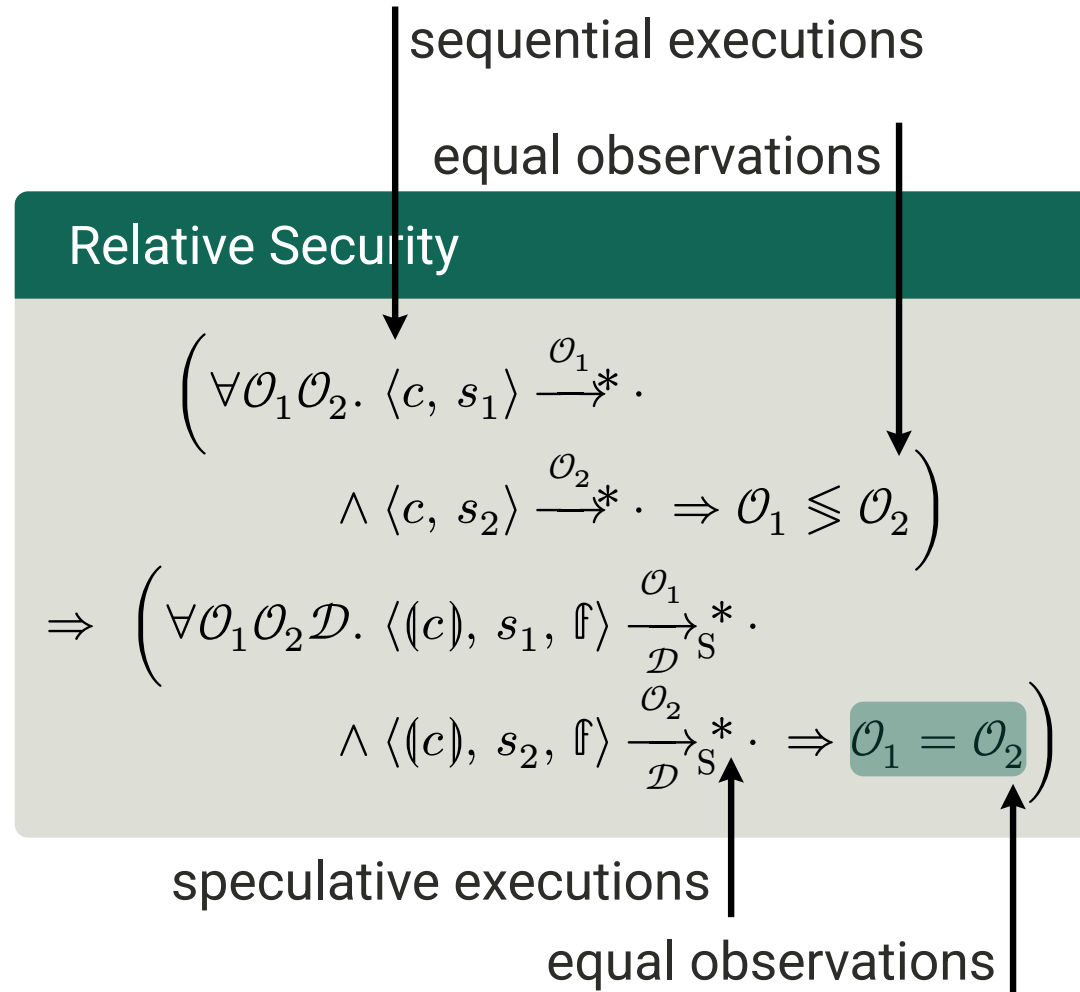
- masks **all sources** of leakage, **always**
 - loads, stores, branch conditions
 - also masks many other leaking instructions
- enforces **Relative Security**



- masks **all sources** of leakage, **always**
 - loads, stores, branch conditions
 - also masks many other leaking instructions
- enforces **Relative Security**



- masks **all sources** of leakage, **always**
 - loads, stores, branch conditions
 - also masks many other leaking instructions
- enforces **Relative Security**



SSLH

- selectively protects secret inputs

FSLH

USLH



SSLH

- selectively protects secret inputs
 - 80% less masking

FSLH

USLH

SSLH

- selectively protects secret inputs
 - 80% less masking

FSLH

USLH

- protects all inputs

SSLH

- selectively protects secret inputs
 - 80% less masking

FSLH

USLH

- protects all inputs
 - high overhead (150%)

SSLH

- selectively protects secret inputs
 - 80% less masking



FSLH

- selectively protects secret inputs

USLH

- protects all inputs
 - high overhead (150%)

SSLH

- selectively protects secret inputs
 - 80% less masking
- applies only to CCT-discipline

FSLH

- selectively protects secret inputs

USLH

- protects all inputs
 - high overhead (150%)

SSLH

- selectively protects secret inputs
 - 80% less masking
- applies only to CCT-discipline
 - enforces SCT security

FSLH

- selectively protects secret inputs

USLH

- protects all inputs
 - high overhead (150%)

SSLH

- selectively protects secret inputs
 - 80% less masking
- applies only to CCT-discipline
 - enforces SCT security

FSLH

- selectively protects secret inputs

USLH

- protects all inputs
 - high overhead (150%)
- protects arbitrary programs

SSLH

- selectively protects secret inputs
 - 80% less masking
- applies only to CCT-discipline
 - enforces SCT security

FSLH

- selectively protects secret inputs

USLH

- protects all inputs
 - high overhead (150%)
- protects arbitrary programs
 - enforces relative security

SSLH

- selectively protects secret inputs
 - 80% less masking
- applies only to CCT-discipline
 - enforces SCT security

FSLH

- selectively protects secret inputs
- protects arbitrary programs
 - enforces relative security

USLH

- protects all inputs
 - high overhead (150%)
- protects arbitrary programs
 - enforces relative security


$$\begin{aligned}(\text{skip}) &\dot{=} \text{skip} \\(\mathbf{x} := e) &\dot{=} \mathbf{x} := e \\(c_1; c_2) &\dot{=} (c_1); (c_2) \\(\text{if } be \text{ then } c_1 \text{ else } c_2) &\dot{=} \text{if } \llbracket be \rrbracket_{\mathbb{B}} \text{ then } \mathbf{b} := \llbracket be \rrbracket_{\mathbb{B}} ? \mathbf{b} : 1; (c_1) \\&\quad \text{else } \mathbf{b} := \llbracket be \rrbracket_{\mathbb{B}} ? 1 : \mathbf{b}; (c_2) \\(\text{while } be \text{ do } c) &\dot{=} \text{while } \llbracket be \rrbracket_{\mathbb{B}} \text{ do} \\&\quad \mathbf{b} := \llbracket be \rrbracket_{\mathbb{B}} ? \mathbf{b} : 1; (c); \\&\quad \mathbf{b} := \llbracket be \rrbracket_{\mathbb{B}} ? 1 : \mathbf{b} \\(\mathbf{X} \leftarrow \mathbf{a}[i]) &\dot{=} \mathbf{X} \leftarrow \mathbf{a}[\llbracket i \rrbracket_{\text{rd}}^{\mathbf{X}}] \\(\mathbf{a}[i] \leftarrow e) &\dot{=} \mathbf{a}[\llbracket i \rrbracket_{\text{wr}}^e] \leftarrow e\end{aligned}$$

$$\begin{aligned}
 (\text{skip}) &\dot{=} \text{skip} \\
 (\mathbf{x} := e) &\dot{=} \mathbf{x} := e \\
 (c_1; c_2) &\dot{=} (c_1); (c_2) \\
 (\text{if } be \text{ then } c_1 \text{ else } c_2) &\dot{=} \text{if } \llbracket be \rrbracket_{\mathbb{B}} \text{ then } \mathbf{b} := \llbracket be \rrbracket_{\mathbb{B}} ? \mathbf{b} : 1; (c_1) \\
 &\quad \text{else } \mathbf{b} := \llbracket be \rrbracket_{\mathbb{B}} ? 1 : \mathbf{b}; (c_2) \\
 (\text{while } be \text{ do } c) &\dot{=} \text{while } \llbracket be \rrbracket_{\mathbb{B}} \text{ do} \\
 &\quad \mathbf{b} := \llbracket be \rrbracket_{\mathbb{B}} ? \mathbf{b} : 1; (c); \\
 &\quad \mathbf{b} := \llbracket be \rrbracket_{\mathbb{B}} ? 1 : \mathbf{b} \\
 (\mathbf{X} \leftarrow \mathbf{a}[i]) &\dot{=} \mathbf{X} \leftarrow \mathbf{a}[\llbracket i \rrbracket_{\text{rd}}^{\mathbf{X}}] \\
 (\mathbf{a}[i] \leftarrow e) &\dot{=} \mathbf{a}[\llbracket i \rrbracket_{\text{wr}}^e] \leftarrow e
 \end{aligned}$$

tracking misspeculation



parametric masking of conditions



$$\begin{aligned}
 (\text{skip}) &\doteq \text{skip} \\
 (\mathbf{x} := e) &\doteq \mathbf{x} := e \\
 (c_1; c_2) &\doteq (c_1); (c_2) \\
 (\text{if } be \text{ then } c_1 \text{ else } c_2) &\doteq \text{if } \llbracket be \rrbracket_{\mathbb{B}} \text{ then } \mathbf{b} := \llbracket be \rrbracket_{\mathbb{B}} ? \mathbf{b} : 1; (c_1) \\
 &\quad \text{else } \mathbf{b} := \llbracket be \rrbracket_{\mathbb{B}} ? 1 : \mathbf{b}; (c_2) \\
 (\text{while } be \text{ do } c) &\doteq \text{while } \llbracket be \rrbracket_{\mathbb{B}} \text{ do} \\
 &\quad \mathbf{b} := \llbracket be \rrbracket_{\mathbb{B}} ? \mathbf{b} : 1; (c); \\
 &\quad \mathbf{b} := \llbracket be \rrbracket_{\mathbb{B}} ? 1 : \mathbf{b} \\
 (\mathbf{X} \leftarrow \mathbf{a}[i]) &\doteq \mathbf{X} \leftarrow \mathbf{a}[\llbracket i \rrbracket_{\text{rd}}^{\mathbf{X}}] \\
 (\mathbf{a}[i] \leftarrow e) &\doteq \mathbf{a}[\llbracket i \rrbracket_{\text{wr}}^e] \leftarrow e
 \end{aligned}$$

$$\begin{aligned}
 (\text{skip}) &\dot{=} \text{skip} \\
 (\mathbf{x} := e) &\dot{=} \mathbf{x} := e \\
 (c_1; c_2) &\dot{=} (c_1); (c_2) \\
 (\text{if } be \text{ then } c_1 \text{ else } c_2) &\dot{=} \text{if } \llbracket be \rrbracket_{\mathbb{B}} \text{ then } \mathbf{b} := \llbracket be \rrbracket_{\mathbb{B}} ? \mathbf{b} : 1; (c_1) \\
 &\quad \text{else } \mathbf{b} := \llbracket be \rrbracket_{\mathbb{B}} ? 1 : \mathbf{b}; (c_2) \\
 (\text{while } be \text{ do } c) &\dot{=} \text{while } \llbracket be \rrbracket_{\mathbb{B}} \text{ do} \\
 &\quad \mathbf{b} := \llbracket be \rrbracket_{\mathbb{B}} ? \mathbf{b} : 1; (c); \\
 &\quad \mathbf{b} := \llbracket be \rrbracket_{\mathbb{B}} ? 1 : \mathbf{b} \\
 (\mathbf{X} \leftarrow \mathbf{a}[i]) &\dot{=} \mathbf{X} \leftarrow \mathbf{a}[\llbracket i \rrbracket_{\text{rd}}^{\mathbf{x}}] \\
 (\mathbf{a}[i] \leftarrow e) &\dot{=} \mathbf{a}[\llbracket i \rrbracket_{\text{wr}}^e] \leftarrow e
 \end{aligned}$$

parametric masking of read indices

$$\begin{aligned}(\text{skip}) &\doteq \text{skip} \\(\mathbf{x} := e) &\doteq \mathbf{x} := e \\(c_1; c_2) &\doteq (c_1); (c_2) \\(\text{if } be \text{ then } c_1 \text{ else } c_2) &\doteq \text{if } \llbracket be \rrbracket_{\mathbb{B}} \text{ then } \mathbf{b} := \llbracket be \rrbracket_{\mathbb{B}} ? \mathbf{b} : 1; (c_1) \\&\quad \text{else } \mathbf{b} := \llbracket be \rrbracket_{\mathbb{B}} ? 1 : \mathbf{b}; (c_2) \\(\text{while } be \text{ do } c) &\doteq \text{while } \llbracket be \rrbracket_{\mathbb{B}} \text{ do} \\&\quad \mathbf{b} := \llbracket be \rrbracket_{\mathbb{B}} ? \mathbf{b} : 1; (c); \\&\quad \mathbf{b} := \llbracket be \rrbracket_{\mathbb{B}} ? 1 : \mathbf{b} \\(\mathbf{X} \leftarrow \mathbf{a}[i]) &\doteq \mathbf{X} \leftarrow \mathbf{a}[\llbracket i \rrbracket_{\text{rd}}^{\mathbf{X}}] \\(\mathbf{a}[i] \leftarrow e) &\doteq \mathbf{a}[\llbracket i \rrbracket_{\text{wr}}^e] \leftarrow e\end{aligned}$$

parametric masking of write indices



SSLH

$$(\text{skip}) \doteq \text{skip}$$

$$(x := e) \doteq x := e$$

$$(c_1; c_2) \doteq (c_1); (c_2)$$

$$(\text{if } be \text{ then } c_1 \text{ else } c_2) \doteq \text{if } \llbracket be \rrbracket_{\mathbb{B}} \text{ then } b := \llbracket be \rrbracket_{\mathbb{B}} ? b : 1; (c_1) \\ \text{else } b := \llbracket be \rrbracket_{\mathbb{B}} ? 1 : b; (c_2)$$

$$(\text{while } be \text{ do } c) \doteq \text{while } \llbracket be \rrbracket_{\mathbb{B}} \text{ do} \\ b := \llbracket be \rrbracket_{\mathbb{B}} ? b : 1; (c); \\ b := \llbracket be \rrbracket_{\mathbb{B}} ? 1 : b$$

$$(X \leftarrow a[i]) \doteq X \leftarrow a[\llbracket i \rrbracket_{\text{rd}}^x]$$

$$(a[i] \leftarrow e) \doteq a[\llbracket i \rrbracket_{\text{wr}}^e] \leftarrow e$$



SSLH

$\llbracket \text{be} \rrbracket_{\mathbb{B}} \doteq \text{be}$

$(\text{skip}) \doteq \text{skip}$

$(\mathbf{x} := e) \doteq \mathbf{x} := e$

$(c_1; c_2) \doteq (c_1); (c_2)$

$(\text{if } \text{be} \text{ then } c_1 \text{ else } c_2) \doteq \text{if } \llbracket \text{be} \rrbracket_{\mathbb{B}} \text{ then } \mathbf{b} := \llbracket \text{be} \rrbracket_{\mathbb{B}} ? \mathbf{b} : 1; (c_1)$
 $\text{else } \mathbf{b} := \llbracket \text{be} \rrbracket_{\mathbb{B}} ? 1 : \mathbf{b}; (c_2)$

$(\text{while } \text{be} \text{ do } c) \doteq \text{while } \llbracket \text{be} \rrbracket_{\mathbb{B}} \text{ do}$

$\mathbf{b} := \llbracket \text{be} \rrbracket_{\mathbb{B}} ? \mathbf{b} : 1; (c);$

$\mathbf{b} := \llbracket \text{be} \rrbracket_{\mathbb{B}} ? 1 : \mathbf{b}$

$(\mathbf{X} \leftarrow \mathbf{a}[i]) \doteq \mathbf{X} \leftarrow \mathbf{a}[\llbracket i \rrbracket_{\text{rd}}^{\mathbf{X}}]$

$(\mathbf{a}[i] \leftarrow e) \doteq \mathbf{a}[\llbracket i \rrbracket_{\text{wr}}^e] \leftarrow e$

SSLH

$$\llbracket \text{be} \rrbracket_{\mathbb{B}} \doteq \text{be}$$

$$\llbracket i \rrbracket_{\text{rd}}^{\mathbf{x}} \doteq \begin{cases} \mathbf{b} == 1 ? 0 : i & \text{if } P(\mathbf{x}) \\ i & \text{otherwise} \end{cases}$$

$$(\text{skip}) \doteq \text{skip}$$

$$(\mathbf{x} := e) \doteq \mathbf{x} := e$$

$$(c_1; c_2) \doteq (c_1); (c_2)$$

$$(\text{if } \text{be} \text{ then } c_1 \text{ else } c_2) \doteq \text{if } \llbracket \text{be} \rrbracket_{\mathbb{B}} \text{ then } \mathbf{b} := \llbracket \text{be} \rrbracket_{\mathbb{B}} ? \mathbf{b} : 1; (c_1) \\ \text{else } \mathbf{b} := \llbracket \text{be} \rrbracket_{\mathbb{B}} ? 1 : \mathbf{b}; (c_2)$$

$$(\text{while } \text{be} \text{ do } c) \doteq \text{while } \llbracket \text{be} \rrbracket_{\mathbb{B}} \text{ do} \\ \mathbf{b} := \llbracket \text{be} \rrbracket_{\mathbb{B}} ? \mathbf{b} : 1; (c); \\ \mathbf{b} := \llbracket \text{be} \rrbracket_{\mathbb{B}} ? 1 : \mathbf{b}$$

$$(\mathbf{X} \leftarrow \mathbf{a}[i]) \doteq \mathbf{X} \leftarrow \mathbf{a}[\llbracket i \rrbracket_{\text{rd}}^{\mathbf{x}}]$$

$$(\mathbf{a}[i] \leftarrow e) \doteq \mathbf{a}[\llbracket i \rrbracket_{\text{wr}}^e] \leftarrow e$$

SSLH

$$\llbracket \text{be} \rrbracket_{\mathbb{B}} \doteq \text{be}$$

$$\llbracket i \rrbracket_{\text{rd}}^{\mathbf{x}} \doteq \begin{cases} \mathbf{b} == 1 ? 0 : i & \text{if } P(\mathbf{x}) \\ i & \text{otherwise} \end{cases}$$

$$\llbracket i \rrbracket_{\text{wr}}^e \doteq \begin{cases} \mathbf{b} == 1 ? 0 : i & \text{if } \neg P(e) \\ i & \text{otherwise} \end{cases}$$

$$(\text{skip}) \doteq \text{skip}$$

$$(\mathbf{x} := e) \doteq \mathbf{x} := e$$

$$(c_1; c_2) \doteq (c_1); (c_2)$$

$$(\text{if be then } c_1 \text{ else } c_2) \doteq \text{if } \llbracket \text{be} \rrbracket_{\mathbb{B}} \text{ then } \mathbf{b} := \llbracket \text{be} \rrbracket_{\mathbb{B}} ? \mathbf{b} : 1; (c_1) \\ \text{else } \mathbf{b} := \llbracket \text{be} \rrbracket_{\mathbb{B}} ? 1 : \mathbf{b}; (c_2)$$

$$(\text{while be do } c) \doteq \text{while } \llbracket \text{be} \rrbracket_{\mathbb{B}} \text{ do} \\ \mathbf{b} := \llbracket \text{be} \rrbracket_{\mathbb{B}} ? \mathbf{b} : 1; (c); \\ \mathbf{b} := \llbracket \text{be} \rrbracket_{\mathbb{B}} ? 1 : \mathbf{b}$$

$$(\mathbf{X} \leftarrow \mathbf{a}[i]) \doteq \mathbf{X} \leftarrow \mathbf{a}[\llbracket i \rrbracket_{\text{rd}}^{\mathbf{x}}]$$

$$(\mathbf{a}[i] \leftarrow e) \doteq \mathbf{a}[\llbracket i \rrbracket_{\text{wr}}^e] \leftarrow e$$

SSLH

$$\llbracket \text{be} \rrbracket_{\mathbb{B}} \doteq \text{be}$$

$$\llbracket i \rrbracket_{\text{rd}}^{\mathbf{x}} \doteq \begin{cases} \mathbf{b} == 1 ? 0 : i & \text{if } P(\mathbf{x}) \\ i & \text{otherwise} \end{cases}$$

$$\llbracket i \rrbracket_{\text{wr}}^e \doteq \begin{cases} \mathbf{b} == 1 ? 0 : i & \text{if } \neg P(e) \\ i & \text{otherwise} \end{cases}$$

USLH

$$(\text{skip}) \doteq \text{skip}$$

$$(\mathbf{x} := e) \doteq \mathbf{x} := e$$

$$(c_1; c_2) \doteq (c_1); (c_2)$$

$$(\text{if be then } c_1 \text{ else } c_2) \doteq \text{if } \llbracket \text{be} \rrbracket_{\mathbb{B}} \text{ then } \mathbf{b} := \llbracket \text{be} \rrbracket_{\mathbb{B}} ? \mathbf{b} : 1; (c_1) \\ \text{else } \mathbf{b} := \llbracket \text{be} \rrbracket_{\mathbb{B}} ? 1 : \mathbf{b}; (c_2)$$

$$(\text{while be do } c) \doteq \text{while } \llbracket \text{be} \rrbracket_{\mathbb{B}} \text{ do} \\ \mathbf{b} := \llbracket \text{be} \rrbracket_{\mathbb{B}} ? \mathbf{b} : 1; (c); \\ \mathbf{b} := \llbracket \text{be} \rrbracket_{\mathbb{B}} ? 1 : \mathbf{b}$$

$$(\mathbf{X} \leftarrow \mathbf{a}[i]) \doteq \mathbf{X} \leftarrow \mathbf{a}[\llbracket i \rrbracket_{\text{rd}}^{\mathbf{x}}]$$

$$(\mathbf{a}[i] \leftarrow e) \doteq \mathbf{a}[\llbracket i \rrbracket_{\text{wr}}^e] \leftarrow e$$

SSLH

$$\llbracket \text{be} \rrbracket_{\mathbb{B}} \doteq \text{be}$$

$$\llbracket i \rrbracket_{\text{rd}}^{\mathbf{x}} \doteq \begin{cases} \mathbf{b} == 1 ? 0 : i & \text{if } P(\mathbf{x}) \\ i & \text{otherwise} \end{cases}$$

$$\llbracket i \rrbracket_{\text{wr}}^e \doteq \begin{cases} \mathbf{b} == 1 ? 0 : i & \text{if } \neg P(e) \\ i & \text{otherwise} \end{cases}$$

USLH

$$\llbracket \text{be} \rrbracket_{\mathbb{B}} \doteq \mathbf{b} == 0 \&\& \text{be}$$

$$(\text{skip}) \doteq \text{skip}$$

$$(\mathbf{x} := e) \doteq \mathbf{x} := e$$

$$(c_1; c_2) \doteq (c_1); (c_2)$$

$$(\text{if be then } c_1 \text{ else } c_2) \doteq \text{if } \llbracket \text{be} \rrbracket_{\mathbb{B}} \text{ then } \mathbf{b} := \llbracket \text{be} \rrbracket_{\mathbb{B}} ? \mathbf{b} : 1; (c_1) \\ \text{else } \mathbf{b} := \llbracket \text{be} \rrbracket_{\mathbb{B}} ? 1 : \mathbf{b}; (c_2)$$

$$(\text{while be do } c) \doteq \text{while } \llbracket \text{be} \rrbracket_{\mathbb{B}} \text{ do}$$

$$\mathbf{b} := \llbracket \text{be} \rrbracket_{\mathbb{B}} ? \mathbf{b} : 1; (c);$$

$$\mathbf{b} := \llbracket \text{be} \rrbracket_{\mathbb{B}} ? 1 : \mathbf{b}$$

$$(\mathbf{X} \leftarrow \mathbf{a}[i]) \doteq \mathbf{X} \leftarrow \mathbf{a}[\llbracket i \rrbracket_{\text{rd}}^{\mathbf{x}}]$$

$$(\mathbf{a}[i] \leftarrow e) \doteq \mathbf{a}[\llbracket i \rrbracket_{\text{wr}}^e] \leftarrow e$$

SSLH

$$\llbracket \text{be} \rrbracket_{\mathbb{B}} \doteq \text{be}$$

$$\llbracket i \rrbracket_{\text{rd}}^{\mathbf{x}} \doteq \begin{cases} \mathbf{b} == 1 ? 0 : i & \text{if } P(\mathbf{x}) \\ i & \text{otherwise} \end{cases}$$

$$\llbracket i \rrbracket_{\text{wr}}^e \doteq \begin{cases} \mathbf{b} == 1 ? 0 : i & \text{if } \neg P(e) \\ i & \text{otherwise} \end{cases}$$

$$(\text{skip}) \doteq \text{skip}$$

$$(\mathbf{x} := e) \doteq \mathbf{x} := e$$

$$(c_1; c_2) \doteq (c_1); (c_2)$$

$$(\text{if be then } c_1 \text{ else } c_2) \doteq \text{if } \llbracket \text{be} \rrbracket_{\mathbb{B}} \text{ then } \mathbf{b} := \llbracket \text{be} \rrbracket_{\mathbb{B}} ? \mathbf{b} : 1; (c_1)$$

$$\text{else } \mathbf{b} := \llbracket \text{be} \rrbracket_{\mathbb{B}} ? 1 : \mathbf{b}; (c_2)$$

$$(\text{while be do } c) \doteq \text{while } \llbracket \text{be} \rrbracket_{\mathbb{B}} \text{ do}$$

$$\mathbf{b} := \llbracket \text{be} \rrbracket_{\mathbb{B}} ? \mathbf{b} : 1; (c);$$

$$\mathbf{b} := \llbracket \text{be} \rrbracket_{\mathbb{B}} ? 1 : \mathbf{b}$$

$$(\mathbf{X} \leftarrow \mathbf{a}[i]) \doteq \mathbf{X} \leftarrow \mathbf{a}[\llbracket i \rrbracket_{\text{rd}}^{\mathbf{x}}]$$

$$(\mathbf{a}[i] \leftarrow e) \doteq \mathbf{a}[\llbracket i \rrbracket_{\text{wr}}^e] \leftarrow e$$

USLH

$$\llbracket \text{be} \rrbracket_{\mathbb{B}} \doteq \mathbf{b} == 0 \&\& \text{be}$$

$$\llbracket i \rrbracket_{\text{rd}}^{\mathbf{x}} \doteq \mathbf{b} == 1 ? 0 : i$$

SSLH

$$\llbracket \text{be} \rrbracket_{\mathbb{B}} \doteq \text{be}$$

$$\llbracket i \rrbracket_{\text{rd}}^{\mathbf{x}} \doteq \begin{cases} \mathbf{b} == 1 ? 0 : i & \text{if } P(\mathbf{x}) \\ i & \text{otherwise} \end{cases}$$

$$\llbracket i \rrbracket_{\text{wr}}^e \doteq \begin{cases} \mathbf{b} == 1 ? 0 : i & \text{if } \neg P(e) \\ i & \text{otherwise} \end{cases}$$

$$(\text{skip}) \doteq \text{skip}$$

$$(\mathbf{x} := e) \doteq \mathbf{x} := e$$

$$(c_1; c_2) \doteq (c_1); (c_2)$$

$$(\text{if be then } c_1 \text{ else } c_2) \doteq \text{if } \llbracket \text{be} \rrbracket_{\mathbb{B}} \text{ then } \mathbf{b} := \llbracket \text{be} \rrbracket_{\mathbb{B}} ? \mathbf{b} : 1; (c_1)$$

$$\text{else } \mathbf{b} := \llbracket \text{be} \rrbracket_{\mathbb{B}} ? 1 : \mathbf{b}; (c_2)$$

$$(\text{while be do } c) \doteq \text{while } \llbracket \text{be} \rrbracket_{\mathbb{B}} \text{ do}$$

$$\mathbf{b} := \llbracket \text{be} \rrbracket_{\mathbb{B}} ? \mathbf{b} : 1; (c);$$

$$\mathbf{b} := \llbracket \text{be} \rrbracket_{\mathbb{B}} ? 1 : \mathbf{b}$$

$$(\mathbf{X} \leftarrow \mathbf{a}[i]) \doteq \mathbf{X} \leftarrow \mathbf{a}[\llbracket i \rrbracket_{\text{rd}}^{\mathbf{x}}]$$

$$(\mathbf{a}[i] \leftarrow e) \doteq \mathbf{a}[\llbracket i \rrbracket_{\text{wr}}^e] \leftarrow e$$

USLH

$$\llbracket \text{be} \rrbracket_{\mathbb{B}} \doteq \mathbf{b} == 0 \&\& \text{be}$$

$$\llbracket i \rrbracket_{\text{rd}}^{\mathbf{x}} \doteq \mathbf{b} == 1 ? 0 : i$$

$$\llbracket i \rrbracket_{\text{wr}}^e \doteq \mathbf{b} == 1 ? 0 : i$$

SSLH

$$\llbracket \text{be} \rrbracket_{\mathbb{B}} \doteq \text{be}$$

$$\llbracket i \rrbracket_{\text{rd}}^{\mathbf{x}} \doteq \begin{cases} \mathbf{b} == 1 ? 0 : i & \text{if } P(\mathbf{x}) \\ i & \text{otherwise} \end{cases}$$

$$\llbracket i \rrbracket_{\text{wr}}^e \doteq \begin{cases} \mathbf{b} == 1 ? 0 : i & \text{if } \neg P(\mathbf{x}) \\ i & \text{otherwise} \end{cases}$$

USLH

$$\llbracket \text{be} \rrbracket_{\mathbb{B}} \doteq \mathbf{b} == 0 \&\& \text{be}$$

$$\llbracket i \rrbracket_{\text{rd}}^{\mathbf{x}} \doteq \mathbf{b} == 1 ? 0 : i$$

$$\llbracket i \rrbracket_{\text{wr}}^e \doteq \mathbf{b} == 1 ? 0 : i$$

FSLH

$$\llbracket \text{be} \rrbracket_{\mathbb{B}} \doteq$$

$$\llbracket i \rrbracket_{\text{rd}}^{\mathbf{x}} \doteq$$

$$\llbracket i \rrbracket_{\text{wr}}^e \doteq$$

SSLH

$$\llbracket \text{be} \rrbracket_{\mathbb{B}} \doteq \text{be}$$

$$\llbracket i \rrbracket_{\text{rd}}^{\text{x}} \doteq \begin{cases} \text{b} == 1 ? 0 : i & \text{if } P(\mathbf{x}) \\ i & \text{otherwise} \end{cases}$$

$$\llbracket i \rrbracket_{\text{wr}}^e \doteq \begin{cases} \text{b} == 1 ? 0 : i & \text{if } \neg P(\mathbf{x}) \\ i & \text{otherwise} \end{cases}$$

USLH

$$\llbracket \text{be} \rrbracket_{\mathbb{B}} \doteq \text{b} == 0 \& \& \text{be}$$

$$\llbracket i \rrbracket_{\text{rd}}^{\text{x}} \doteq \text{b} == 1 ? 0 : i$$

$$\llbracket i \rrbracket_{\text{wr}}^e \doteq \text{b} == 1 ? 0 : i$$

FSLH

$$\llbracket \text{be} \rrbracket_{\mathbb{B}} \doteq \begin{cases} \text{b} == 0 \& \& \text{be} \\ \end{cases}$$

$$\llbracket i \rrbracket_{\text{rd}}^{\text{x}} \doteq$$

$$\llbracket i \rrbracket_{\text{wr}}^e \doteq$$

SSLH

$$\llbracket \text{be} \rrbracket_{\mathbb{B}} \doteq \text{be}$$

$$\llbracket i \rrbracket_{\text{rd}}^{\mathbf{x}} \doteq \begin{cases} \mathbf{b} == 1 ? 0 : i & \text{if } P(\mathbf{x}) \\ i & \text{otherwise} \end{cases}$$

$$\llbracket i \rrbracket_{\text{wr}}^e \doteq \begin{cases} \mathbf{b} == 1 ? 0 : i & \text{if } \neg P(\mathbf{x}) \\ i & \text{otherwise} \end{cases}$$

USLH

$$\llbracket \text{be} \rrbracket_{\mathbb{B}} \doteq \mathbf{b} == 0 \& \& \text{be}$$

$$\llbracket i \rrbracket_{\text{rd}}^{\mathbf{x}} \doteq \mathbf{b} == 1 ? 0 : i$$

$$\llbracket i \rrbracket_{\text{wr}}^e \doteq \mathbf{b} == 1 ? 0 : i$$

FSLH

$$\llbracket \text{be} \rrbracket_{\mathbb{B}} \doteq \begin{cases} \mathbf{b} == 0 \& \& \text{be} & \text{if } \neg P(\text{be}) \\ \text{be} & \text{otherwise} \end{cases}$$

$$\llbracket i \rrbracket_{\text{rd}}^{\mathbf{x}} \doteq$$

$$\llbracket i \rrbracket_{\text{wr}}^e \doteq$$

SSLH

$$\llbracket be \rrbracket_{\mathbb{B}} \doteq be$$

$$\llbracket i \rrbracket_{\text{rd}}^{\mathbf{x}} \doteq \begin{cases} \mathbf{b} == 1 ? 0 : i & \text{if } P(\mathbf{x}) \\ i & \text{otherwise} \end{cases}$$

$$\llbracket i \rrbracket_{\text{wr}}^e \doteq \begin{cases} \mathbf{b} == 1 ? 0 : i & \text{if } \neg P(e) \\ i & \text{otherwise} \end{cases}$$

USLH

$$\llbracket be \rrbracket_{\mathbb{B}} \doteq \mathbf{b} == 0 \& \& be$$

$$\llbracket i \rrbracket_{\text{rd}}^{\mathbf{x}} \doteq \mathbf{b} == 1 ? 0 : i$$

$$\llbracket i \rrbracket_{\text{wr}}^e \doteq \mathbf{b} == 1 ? 0 : i$$

FSLH

$$\llbracket be \rrbracket_{\mathbb{B}} \doteq \begin{cases} \mathbf{b} == 0 \& \& be & \text{if } \neg P(be) \\ be & \text{otherwise} \end{cases}$$

$$\llbracket i \rrbracket_{\text{rd}}^{\mathbf{x}} \doteq \begin{cases} \mathbf{b} == 1 ? 0 : i & \text{if } P(\mathbf{x}) \\ i & \text{otherwise} \end{cases}$$

$$\llbracket i \rrbracket_{\text{wr}}^e \doteq \begin{cases} \mathbf{b} == 1 ? 0 : i & \text{if } \neg P(e) \\ i & \text{otherwise} \end{cases}$$

SSLH

$$\llbracket \text{be} \rrbracket_{\mathbb{B}} \doteq \text{be}$$

$$\llbracket i \rrbracket_{\text{rd}}^{\mathbf{x}} \doteq \begin{cases} \mathbf{b} == 1 ? 0 : i & \text{if } P(\mathbf{x}) \\ i & \text{otherwise} \end{cases}$$

$$\llbracket i \rrbracket_{\text{wr}}^e \doteq \begin{cases} \mathbf{b} == 1 ? 0 : i & \text{if } \neg P(e) \\ i & \text{otherwise} \end{cases}$$

USLH

$$\llbracket \text{be} \rrbracket_{\mathbb{B}} \doteq \mathbf{b} == 0 \&\& \text{be}$$

$$\llbracket i \rrbracket_{\text{rd}}^{\mathbf{x}} \doteq \mathbf{b} == 1 ? 0 : i$$

$$\llbracket i \rrbracket_{\text{wr}}^e \doteq \mathbf{b} == 1 ? 0 : i$$

FSLH

$$\llbracket \text{be} \rrbracket_{\mathbb{B}} \doteq \begin{cases} \mathbf{b} == 0 \&\& \text{be} & \text{if } \neg P(\text{be}) \\ \text{be} & \text{otherwise} \end{cases}$$

$$\llbracket i \rrbracket_{\text{rd}}^{\mathbf{x}} \doteq \begin{cases} \mathbf{b} == 1 ? 0 : i & \text{if } P(\mathbf{x}) \vee \neg P(i) \\ i & \text{otherwise} \end{cases}$$

$$\llbracket i \rrbracket_{\text{wr}}^e \doteq \begin{cases} \mathbf{b} == 1 ? 0 : i & \text{if } \neg P(e) \vee \neg P(i) \\ i & \text{otherwise} \end{cases}$$

SSLH

$$\llbracket \text{be} \rrbracket_{\mathbb{B}} \doteq \text{be}$$

$$\llbracket i \rrbracket_{\text{rd}}^{\mathbf{x}} \doteq \begin{cases} \mathbf{b} == 1 ? 0 : i & \text{if } P(\mathbf{x}) \\ i & \text{otherwise} \end{cases}$$

$$\llbracket i \rrbracket_{\text{wr}}^e \doteq \begin{cases} \mathbf{b} == 1 ? 0 : i & \text{if } \neg P(e) \\ i & \text{otherwise} \end{cases}$$

USLH

$$\llbracket \text{be} \rrbracket_{\mathbb{B}} \doteq \mathbf{b} == 0 \&\& \text{be}$$

$$\llbracket i \rrbracket_{\text{rd}}^{\mathbf{x}} \doteq \mathbf{b} == 1 ? 0 : i$$

$$\llbracket i \rrbracket_{\text{wr}}^e \doteq \mathbf{b} == 1 ? 0 : i$$

FSLH

$$\llbracket \text{be} \rrbracket_{\mathbb{B}} \doteq \begin{cases} \mathbf{b} == 0 \&\& \text{be} & \text{if } \neg P(\text{be}) \\ \text{be} & \text{otherwise} \end{cases}$$

$$\llbracket i \rrbracket_{\text{rd}}^{\mathbf{x}} \doteq \begin{cases} \mathbf{b} == 1 ? 0 : i & \text{if } P(\mathbf{x}) \vee \neg P(i) \\ i & \text{otherwise} \end{cases}$$

$$\llbracket i \rrbracket_{\text{wr}}^e \doteq \begin{cases} \mathbf{b} == 1 ? 0 : i & \text{if } \neg P(e) \vee \neg P(i) \\ i & \text{otherwise} \end{cases}$$

SSLH

$$\llbracket \text{be} \rrbracket_{\mathbb{B}} \doteq \text{be}$$

$$\llbracket i \rrbracket_{\text{rd}}^{\mathbf{x}} \doteq \begin{cases} \mathbf{b} == 1 ? 0 : i & \text{if } P(\mathbf{x}) \\ i & \text{otherwise} \end{cases}$$

$$\llbracket i \rrbracket_{\text{wr}}^e \doteq \begin{cases} \mathbf{b} == 1 ? 0 : i & \text{if } \neg P(e) \\ i & \text{otherwise} \end{cases}$$

USLH

$$\llbracket \text{be} \rrbracket_{\mathbb{B}} \doteq \mathbf{b} == 0 \&\& \text{be}$$

$$\llbracket i \rrbracket_{\text{rd}}^{\mathbf{x}} \doteq \mathbf{b} == 1 ? 0 : i$$

$$\llbracket i \rrbracket_{\text{wr}}^e \doteq \mathbf{b} == 1 ? 0 : i$$

FSLH

$$\llbracket \text{be} \rrbracket_{\mathbb{B}} \doteq \begin{cases} \mathbf{b} == 0 \&\& \text{be} & \text{if } \neg P(\text{be}) \\ \text{be} & \text{otherwise} \end{cases}$$

$$\llbracket i \rrbracket_{\text{rd}}^{\mathbf{x}} \doteq \begin{cases} \mathbf{b} == 1 ? 0 : i & \text{if } P(\mathbf{x}) \vee \neg P(i) \\ i & \text{otherwise} \end{cases}$$

$$\llbracket i \rrbracket_{\text{wr}}^e \doteq \begin{cases} \mathbf{b} == 1 ? 0 : i & \text{if } \neg P(e) \vee \neg P(i) \\ i & \text{otherwise} \end{cases}$$



- Simplest formalisation: standard IFC type system (Volpano, Irvine, and Smith 1996)



- Simplest formalisation: standard IFC type system (Volpano, Irvine, and Smith 1996)
⚠ keeps the security levels of variables fixed, so not all programs are well-typed



- Simplest formalisation: standard IFC type system (Volpano, Irvine, and Smith 1996)
⚠ keeps the security levels of variables fixed, so not all programs are well-typed
- Complete protection: Flow-sensitive static analysis (Hunt and Sands 2006)

- Simplest formalisation: standard IFC type system (Volpano, Irvine, and Smith 1996)
⚠ keeps the security levels of variables fixed, so not all programs are well-typed
- Complete protection: Flow-sensitive static analysis (Hunt and Sands 2006)
 - Given initial labeling, compute a final labeling



- Simplest formalisation: standard IFC type system (Volpano, Irvine, and Smith 1996)
⚠ keeps the security levels of variables fixed, so not all programs are well-typed
- Complete protection: Flow-sensitive static analysis (Hunt and Sands 2006)
 - Given initial labeling, compute a final labeling
 - Overapproximation:

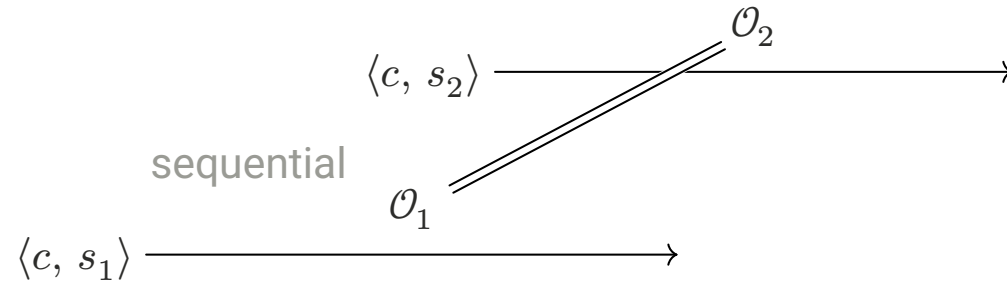


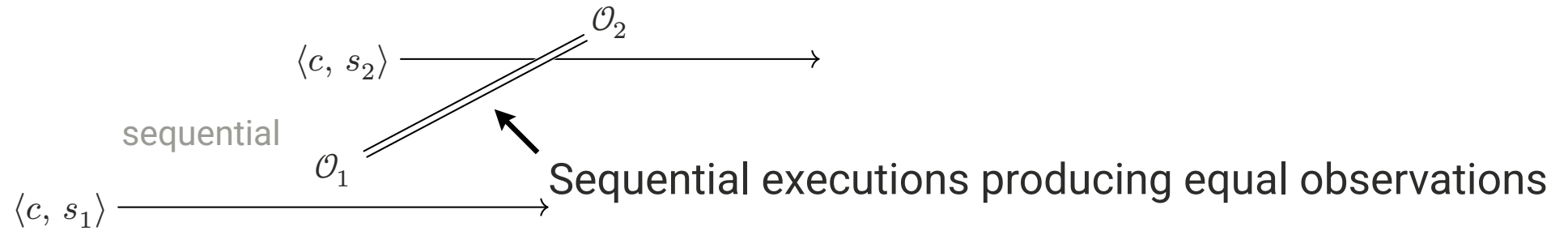
- Simplest formalisation: standard IFC type system (Volpano, Irvine, and Smith 1996)
⚠ keeps the security levels of variables fixed, so not all programs are well-typed
- Complete protection: Flow-sensitive static analysis (Hunt and Sands 2006)
 - Given initial labeling, compute a final labeling
 - Overapproximation:
 - join after branches

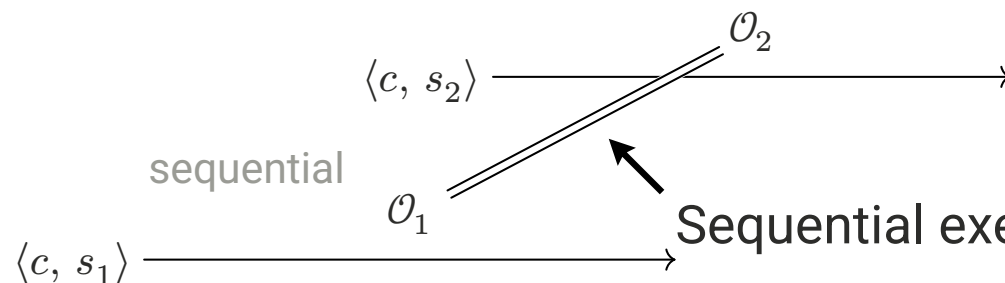


- Simplest formalisation: standard IFC type system (Volpano, Irvine, and Smith 1996)
⚠ keeps the security levels of variables fixed, so not all programs are well-typed
- Complete protection: Flow-sensitive static analysis (Hunt and Sands 2006)
 - Given initial labeling, compute a final labeling
 - Overapproximation:
 - join after branches
 - fixpoint for loops

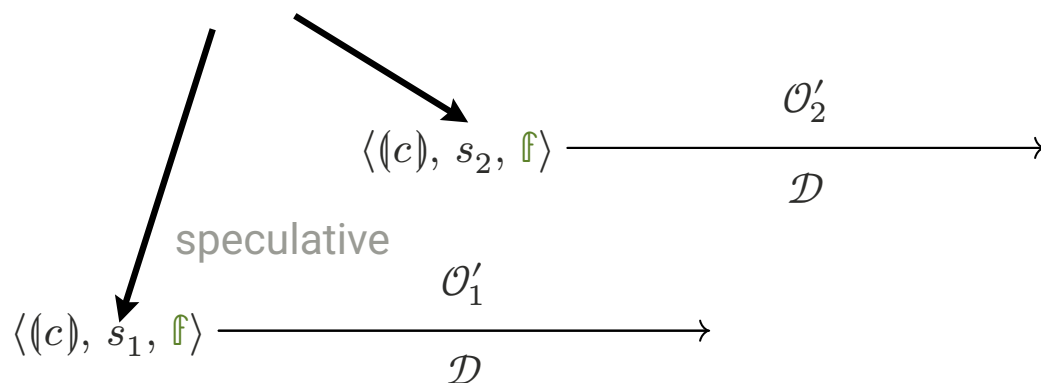
- Simplest formalisation: standard IFC type system (Volpano, Irvine, and Smith 1996)
⚠ keeps the security levels of variables fixed, so not all programs are well-typed
- Complete protection: Flow-sensitive static analysis (Hunt and Sands 2006)
 - Given initial labeling, compute a final labeling
 - Overapproximation:
 - join after branches
 - fixpoint for loops
 - Annotate program with security levels of expressions

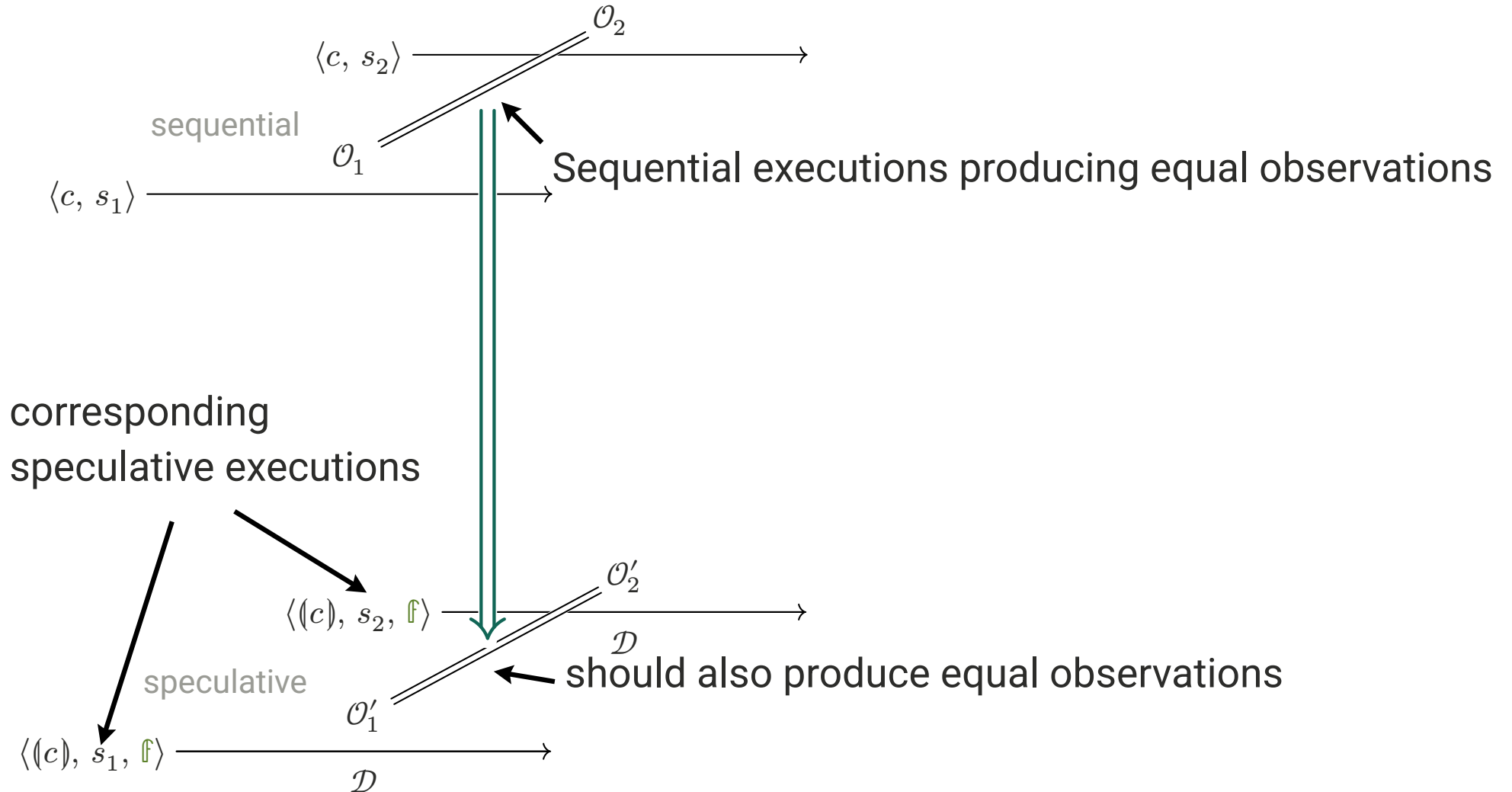


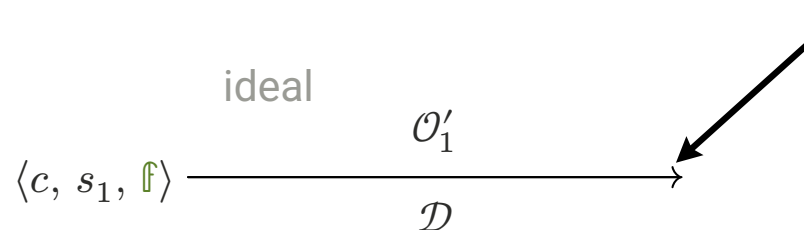
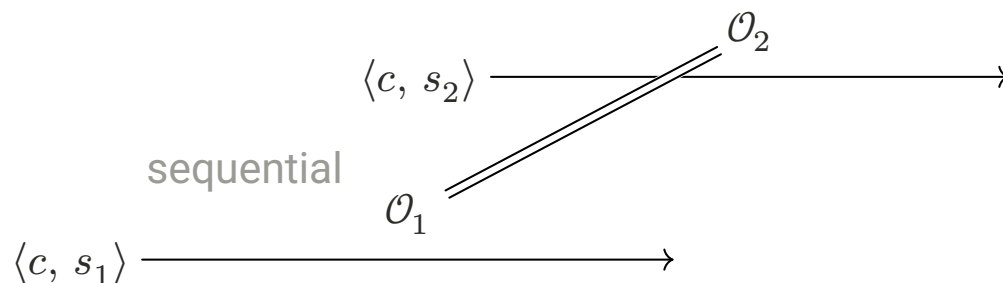




corresponding
speculative executions

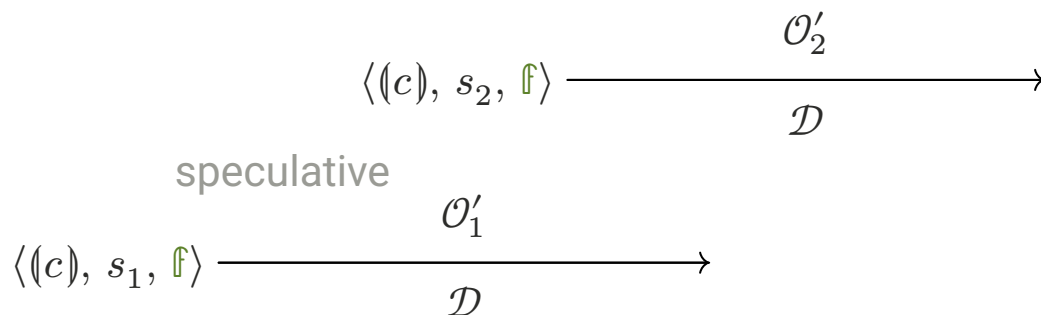


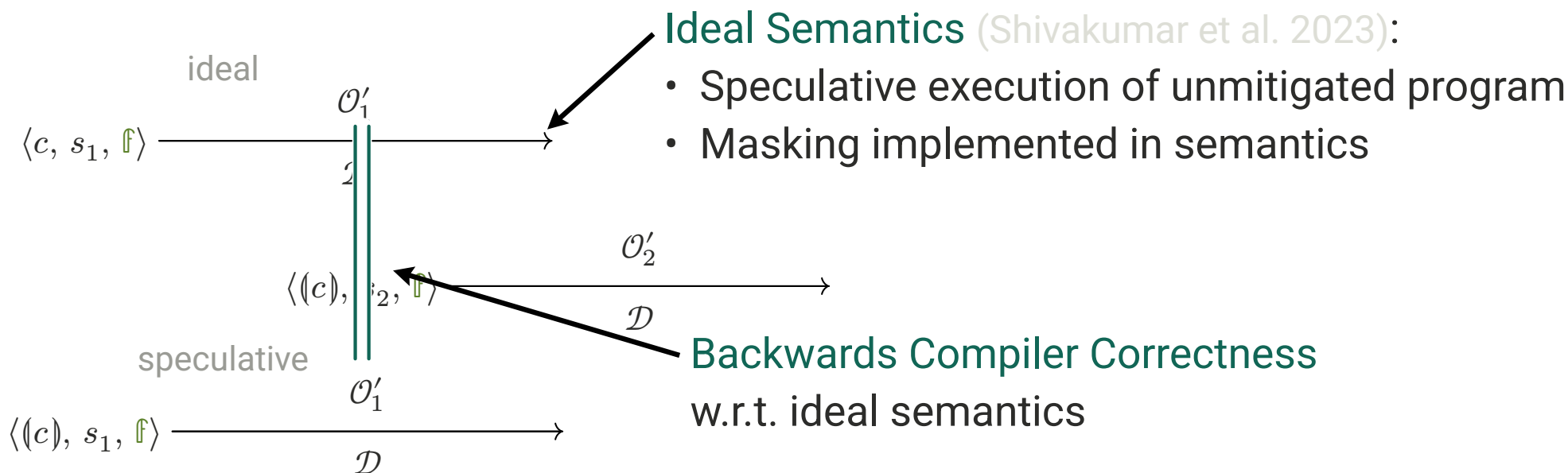
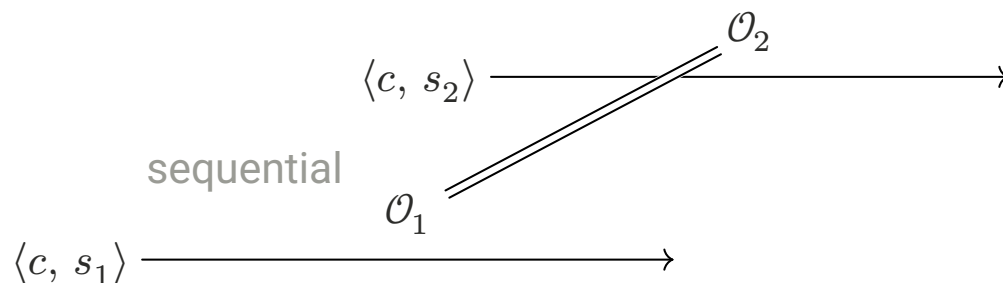


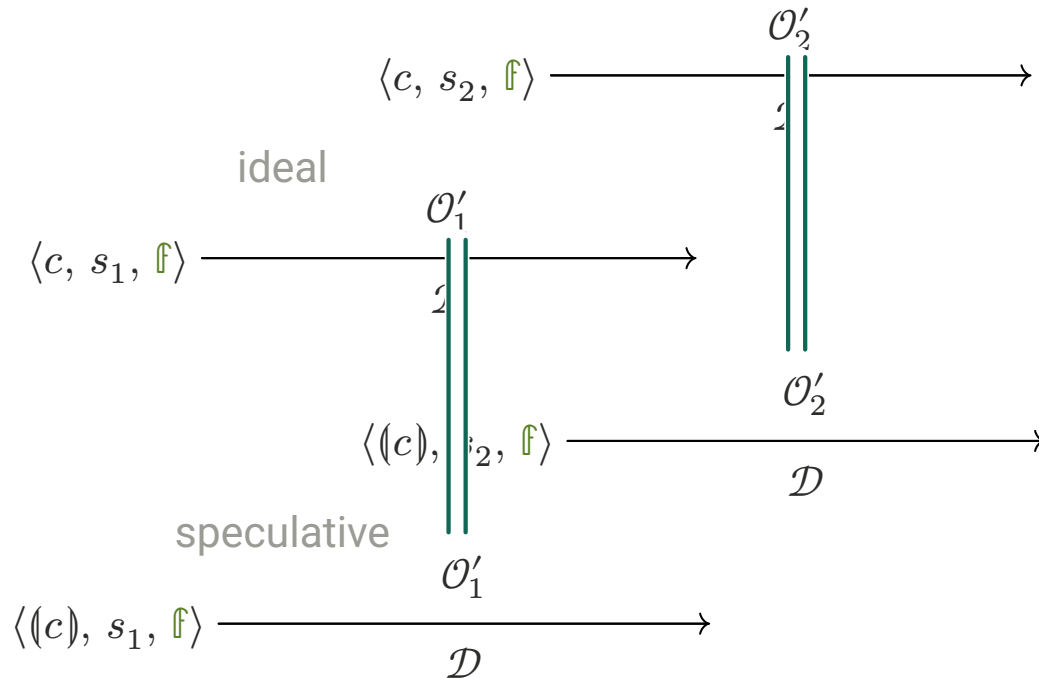
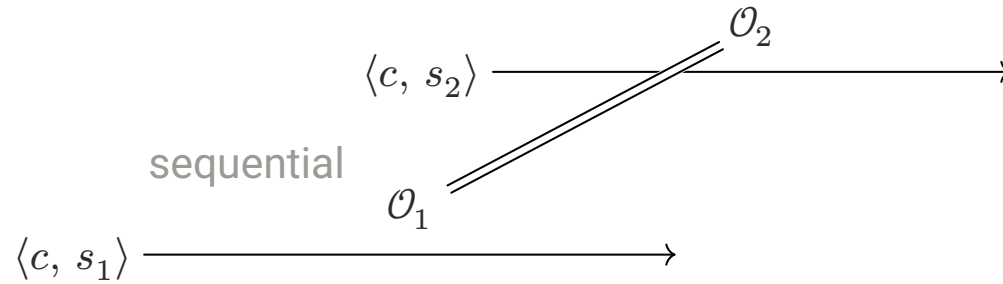


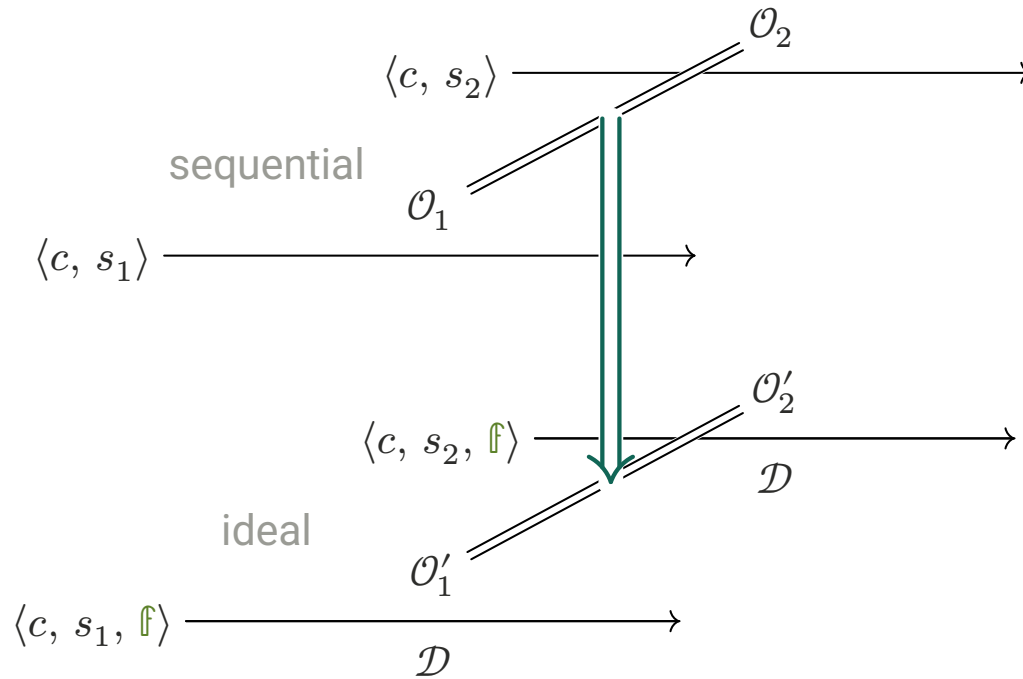
Ideal Semantics (Shivakumar et al. 2023):

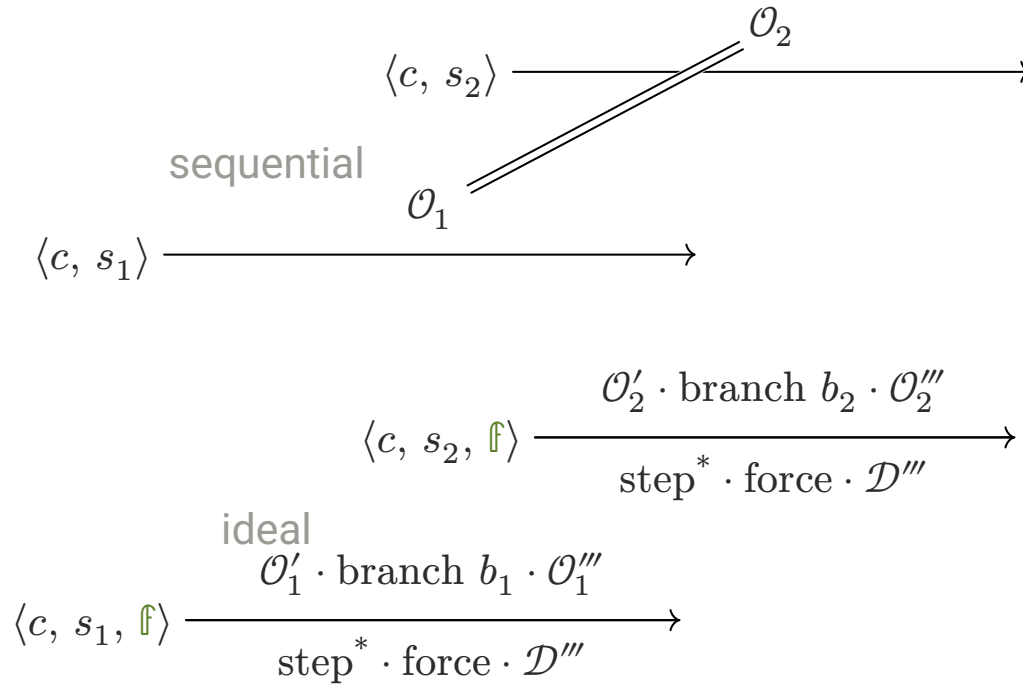
- Speculative execution of unmitigated program
- Masking implemented in semantics

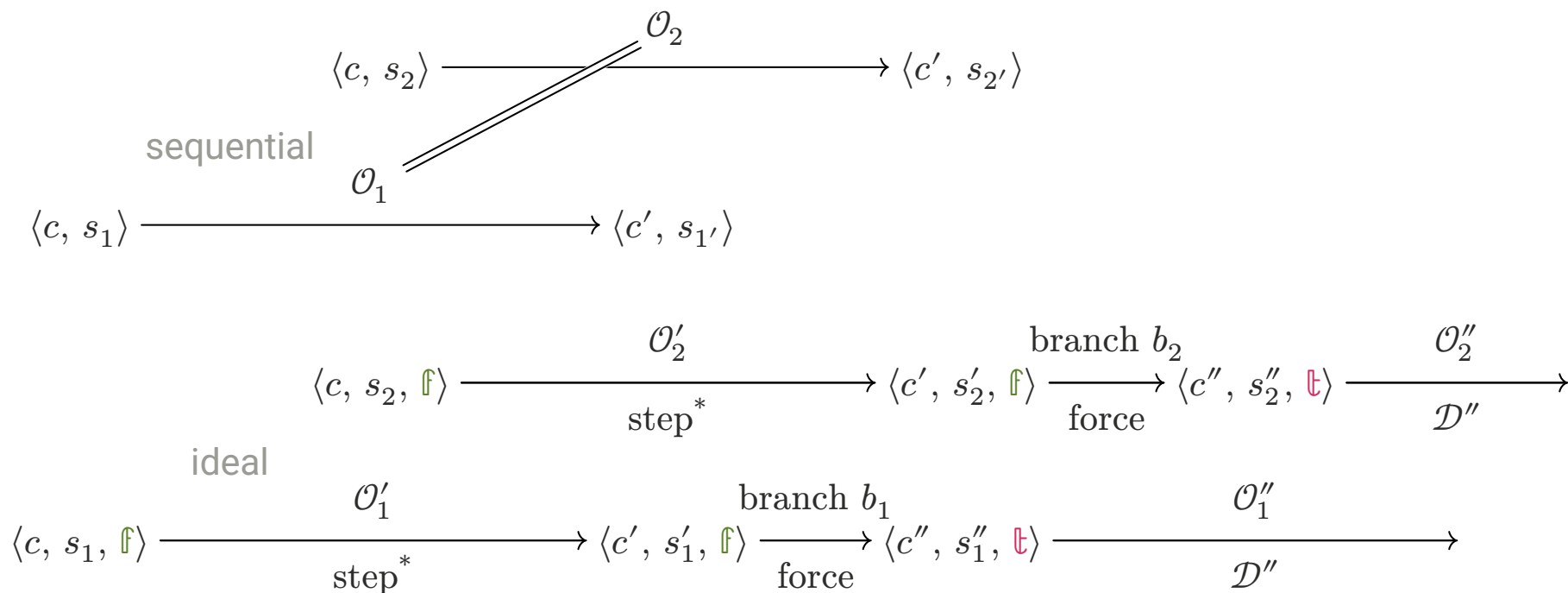


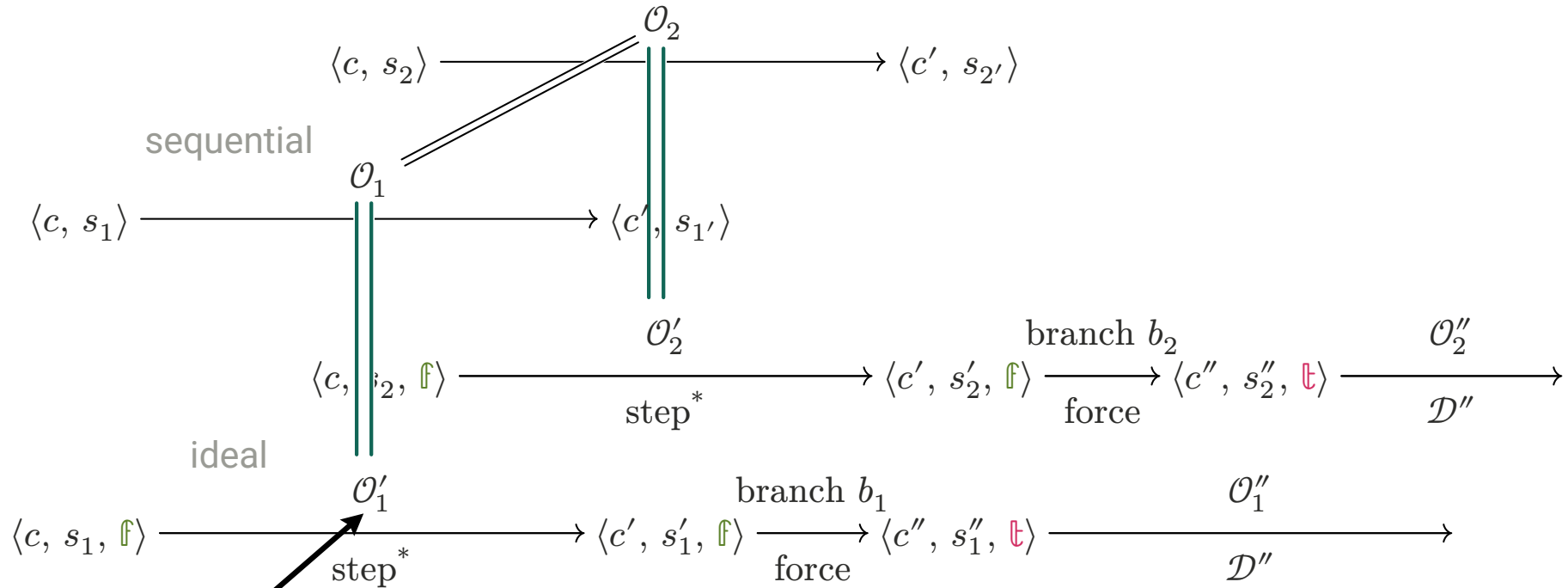




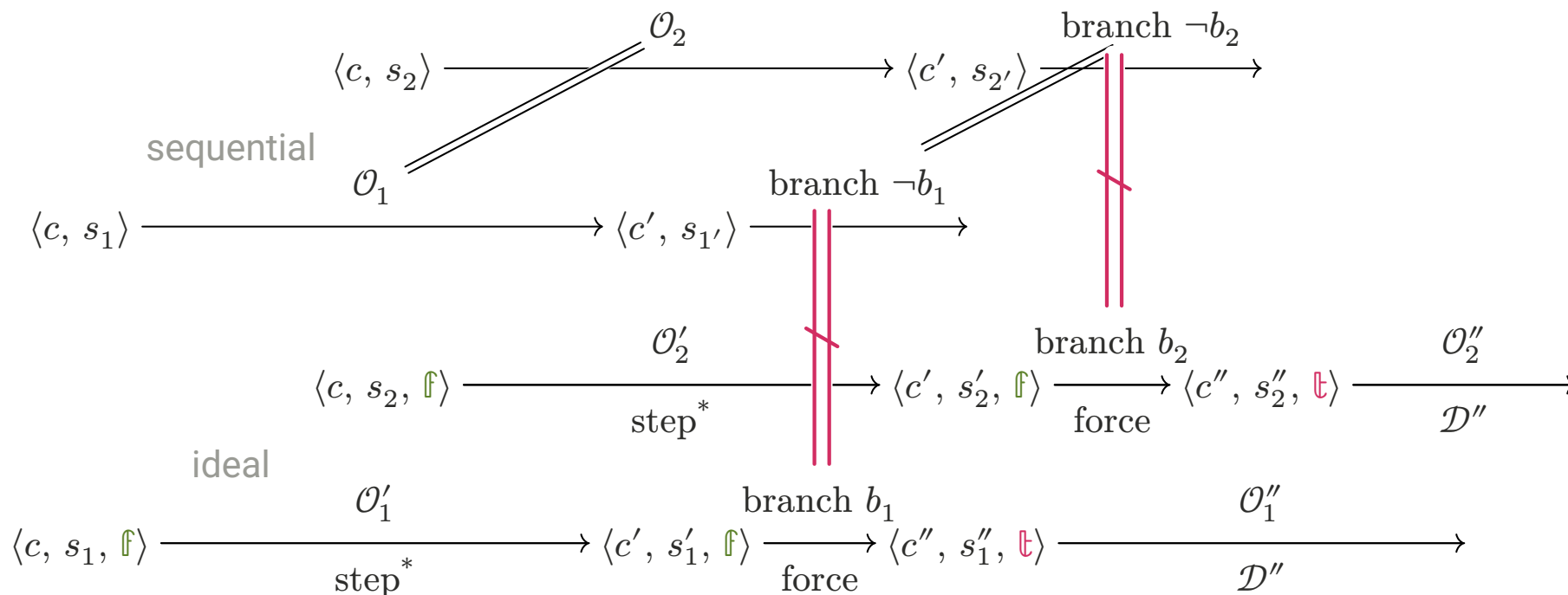




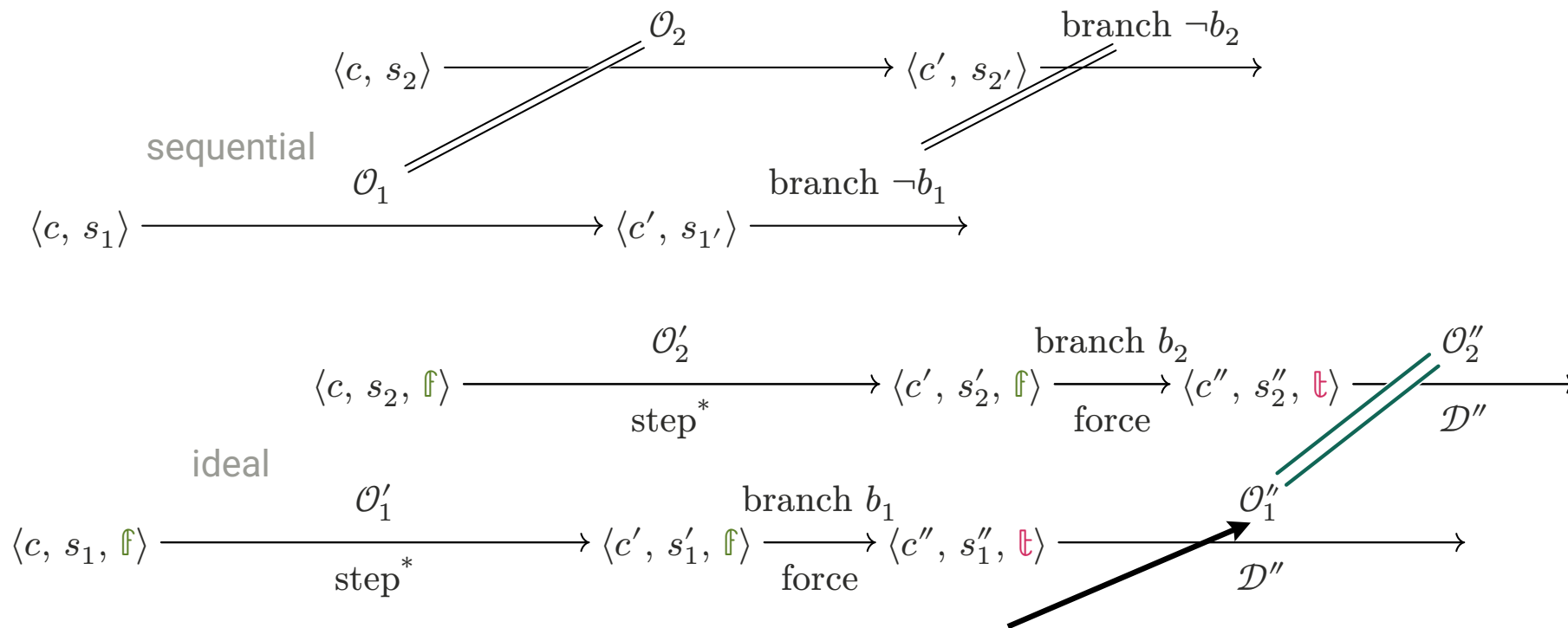


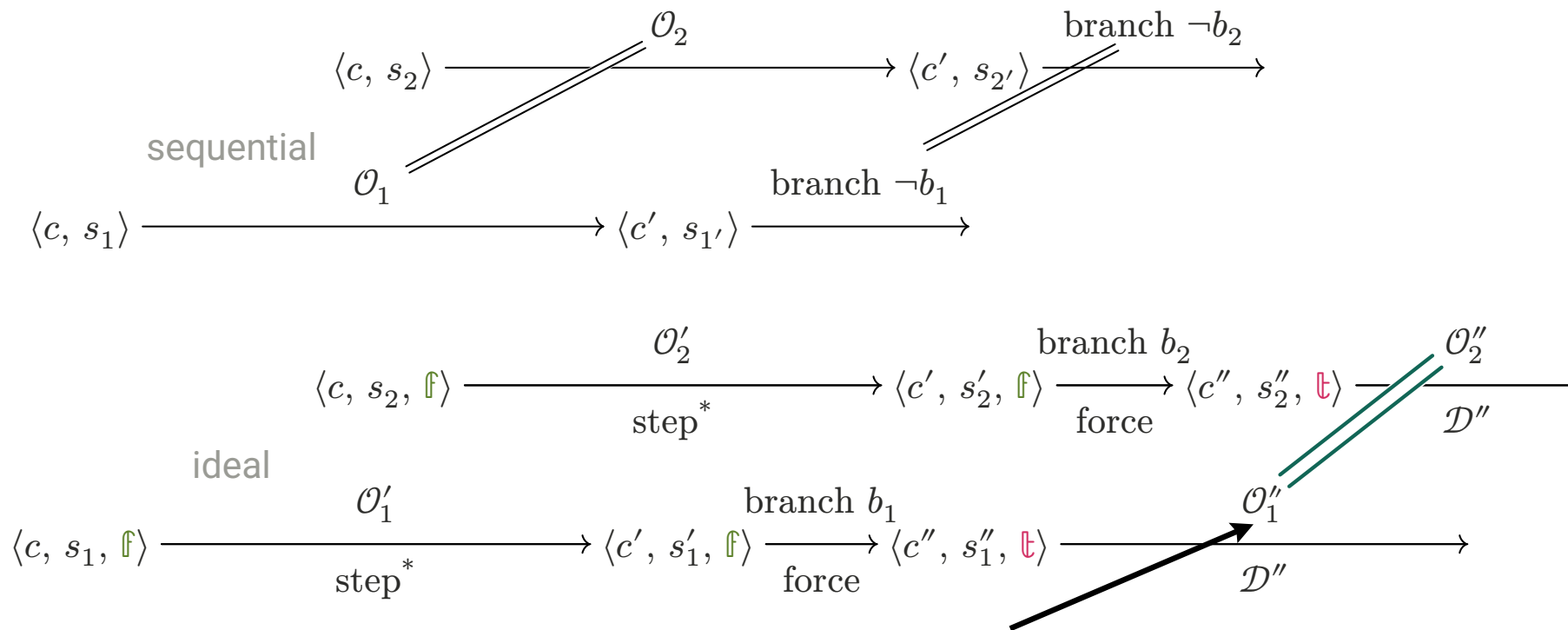


Same as sequential execution



Opposite branch as
sequential execution

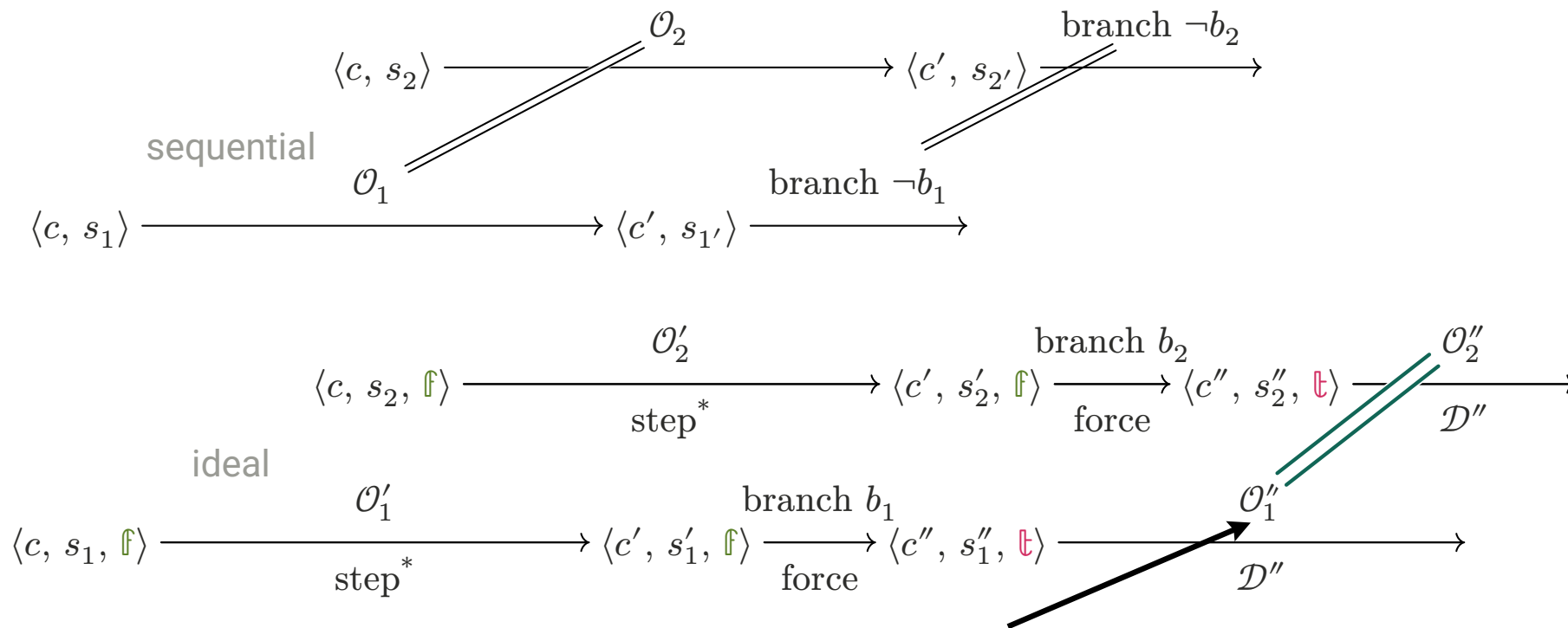




Unwinding:

Observations depend only on directives

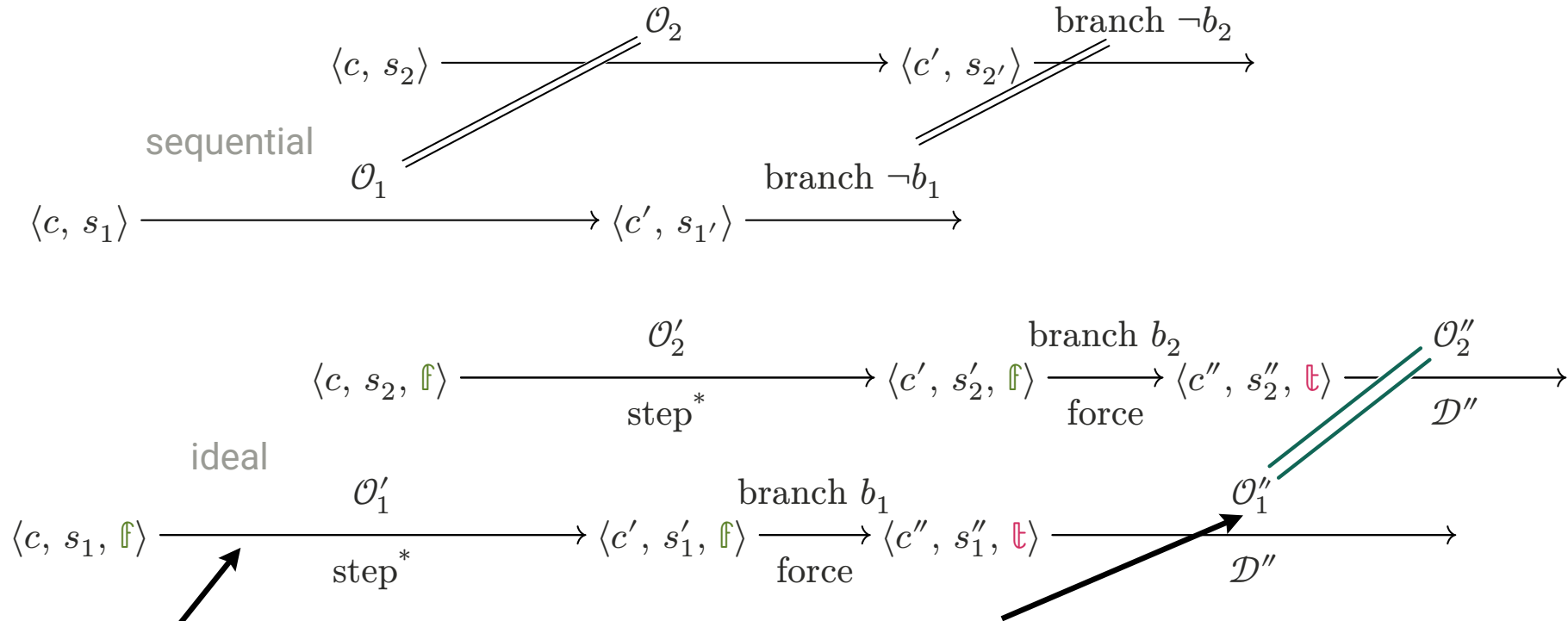
- public values are equal

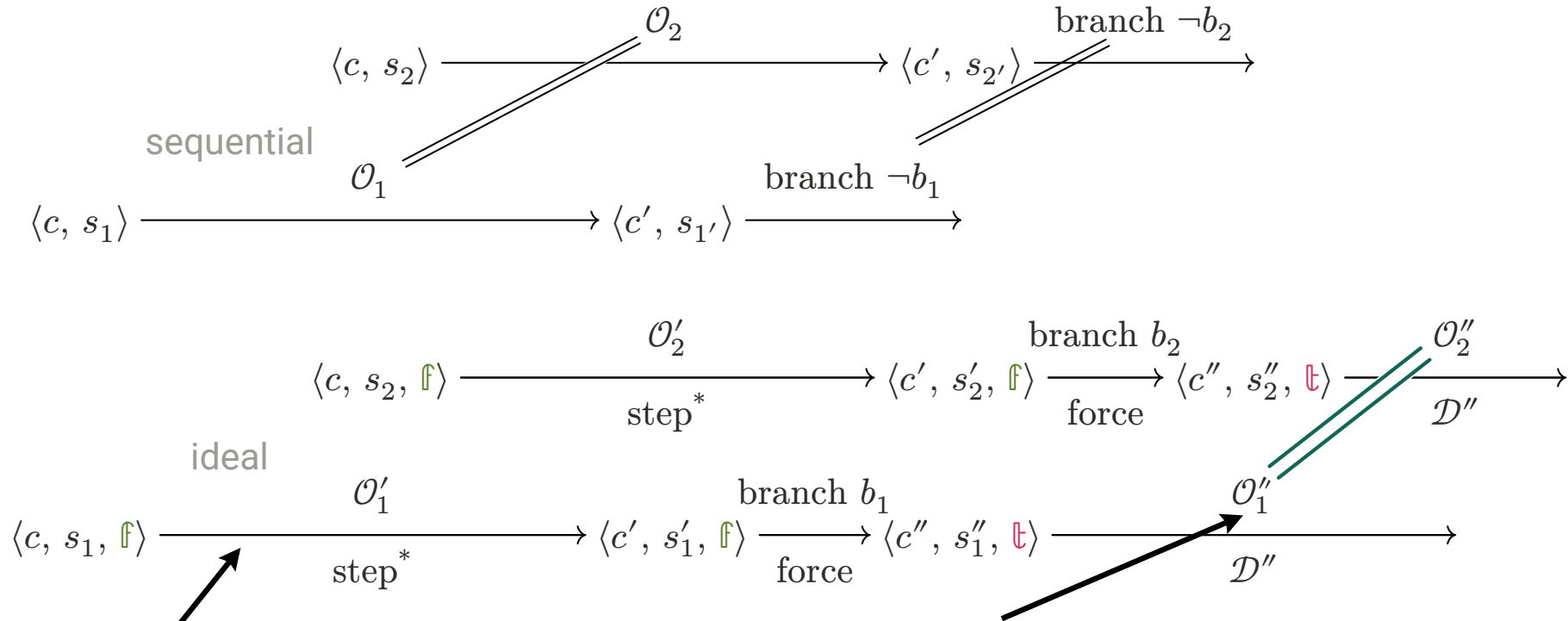


Unwinding:

Observations depend only on directives

- public values are equal
- secret values are masked







- Flexible SLH combines the benefits of Selective and Ultimate SLH



- Flexible SLH combines the benefits of Selective and Ultimate SLH
 - same level of protection as USLH

- Flexible SLH combines the benefits of Selective and Ultimate SLH
 - same level of protection as USLH
 - no overhead compared to SSLH on constant-time code

- Flexible SLH combines the benefits of Selective and Ultimate SLH
 - same level of protection as USLH
 - no overhead compared to SSLH on constant-time code
- First machine-checked proofs of SLH mitigations

- Flexible SLH combines the benefits of Selective and Ultimate SLH
 - same level of protection as USLH
 - no overhead compared to SSLH on constant-time code
- First machine-checked proofs of SLH mitigations
 - proofs of SSLH and USLH obtained as corollaries



- Practical implementation and evaluation of FSLH

- Practical implementation and evaluation of FSLH
 - at what level should analysis be performed?

- Practical implementation and evaluation of FSLH
 - at what level should analysis be performed?
 - preservation by other compilation passes?

- Practical implementation and evaluation of FSLH
 - at what level should analysis be performed?
 - preservation by other compilation passes?
- Investigation of other LLVM SLH implementations



- Practical implementation and evaluation of FSLH
 - at what level should analysis be performed?
 - preservation by other compilation passes?
- Investigation of other LLVM SLH implementations
 - too much complexity for fully mechanized proofs



- Practical implementation and evaluation of FSLH
 - at what level should analysis be performed?
 - preservation by other compilation passes?
- Investigation of other LLVM SLH implementations
 - too much complexity for fully mechanized proofs
 - Property-based testing as a pragmatic compromise

- Carruth, Chandler. 2018. “Speculative Load Hardening: A Spectre Variant #1 Mitigation Technique”. September 2018. <https://llvm.org/docs/SpeculativeLoadHardening.html>.
- Hunt, Sebastian, and David Sands. 2006. “On Flow-Sensitive Security Types”. In *Proceedings of the 33rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2006, Charleston, South Carolina, USA, January 11-13, 2006*, edited by J. Gregory Morrisett and Simon L. Peyton Jones, 79–90. ACM. <https://doi.org/10.1145/1111037.1111045>.
- Shivakumar, Basavesh Ammanaghatta, Jack Barnes, Gilles Barthe, Sunjay Cauligi, Chitchanok Chuengsatiansup, Daniel Genkin, Sioli O'Connell, Peter Schwabe, Rui Qi Sim, and Yuval Yarom. 2023. “Spectre Declassified: Reading from the Right Place at the Wrong Time”. In *44th IEEE Symposium on Security and Privacy, SP*, 1753–70. IEEE. <https://doi.org/10.1109/SP46215.2023.10179355>.

- Volpano, Dennis M., Cynthia E. Irvine, and Geoffrey Smith. 1996. “A Sound Type System for Secure Flow Analysis”. *J. Comput. Secur.* 4 (2/3): 167–88. <https://doi.org/10.3233/JCS-1996-42-304>.
- Zhang, Zhiyuan, Gilles Barthe, Chitchanok Chuengsatiansup, Peter Schwabe, and Yuval Yarom. 2023. “Ultimate SLH: Taking Speculative Load Hardening to the Next Level”. In *32nd USENIX Security Symposium*, edited by Joseph A. Calandrino and Carmela Troncoso, 7125–42. USENIX Association. <https://www.usenix.org/conference/usenixsecurity23/presentation/zhang-zhiyuan-slh>.

$$\text{SPEC_ASGN} \frac{v = \llbracket ae \rrbracket_\rho}{\langle X := ae, \rho, \mu, b \rangle \xrightarrow[\bullet]{s} \langle \text{skip}, [X \mapsto v] \rho, \mu, b \rangle}$$

$$\text{SPEC_SEQ_STEP} \frac{\langle c_1, \rho, \mu, b \rangle \xrightarrow[d]{o} \langle c'_1, \rho', \mu', b' \rangle}{\langle c_1; c_2, \rho, \mu, b \rangle \xrightarrow[d]{o} \langle c'_1; c_2, \rho', \mu', b' \rangle}$$

$$\text{SPEC_WHILE} \frac{c_{\text{while}} = \text{while } be \text{ do } c}{\langle c_{\text{while}}, \rho, \mu, b \rangle \xrightarrow[\bullet]{s} \langle \text{if } be \text{ then } c; c_{\text{while}} \text{ else skip}, \rho, \mu, b \rangle}$$

$$\text{SPEC_SEQ_SKIP} \frac{}{\langle \text{skip}; c, \rho, \mu, b \rangle \xrightarrow[\bullet]{s} \langle c, \rho, \mu, b \rangle}$$

$$\text{SPEC_IF} \frac{b' = \llbracket be \rrbracket_\rho}{\langle \text{if } be \text{ then } c_{\text{T}} \text{ else } c_{\text{F}}, \rho, \mu, b \rangle \xrightarrow[\text{step}]{\text{branch } b'} \langle c_{b'}, \rho, \mu, b \rangle}$$

$$\text{SPEC_IF_FORCE} \frac{b' = \llbracket be \rrbracket_\rho}{\langle \text{if } be \text{ then } c_{\text{T}} \text{ else } c_{\text{F}}, \rho, \mu, b \rangle \xrightarrow[\text{force}]{\text{branch } b'} \langle c_{\neg b'}, \rho, \mu, \mathbb{T} \rangle}$$

$$\text{SPEC_READ} \frac{i = \llbracket ie \rrbracket_\rho \quad v = \llbracket a[i] \rrbracket_\mu \quad i < |a|_\mu}{\langle X \leftarrow a[ie], \rho, \mu, b \rangle \xrightarrow[\text{step}]{\text{read } a^i} \langle \text{skip}, [X \mapsto v] \rho, \mu, b \rangle}$$

$$\text{SPEC_READ_FORCE} \frac{i = \llbracket ie \rrbracket_\rho \quad v = \llbracket b[j] \rrbracket_\mu \quad i \geq |a|_\mu \quad j < |b|_\mu}{\langle X \leftarrow a[ie], \rho, \mu, \mathbb{T} \rangle \xrightarrow[\text{load } b^j]{\text{read } a^i} \langle \text{skip}, [X \mapsto v] \rho, \mu, \mathbb{T} \rangle}$$

$$\text{SPEC_WRITE} \frac{i = \llbracket ie \rrbracket_\rho \quad v = \llbracket ae \rrbracket_\rho \quad i < |a|_\mu}{\langle a[ie] \leftarrow ae, \rho, \mu, b \rangle \xrightarrow[\text{step}]{\text{write } a^i} \langle \text{skip}, \rho, [a[i] \mapsto v] \mu, b \rangle}$$

$$\text{SPEC_WRITE_FORCE} \frac{i = \llbracket ie \rrbracket_\rho \quad v = \llbracket ae \rrbracket_\rho \quad i \geq |a|_\mu \quad j < |b|_\mu}{\langle a[ie] \leftarrow ae, \rho, \mu, \mathbb{T} \rangle \xrightarrow[\text{store } b^j]{\text{write } a^i} \langle \text{skip}, \rho, [b[j] \mapsto v] \mu, \mathbb{T} \rangle}$$

$$\begin{array}{c}
 \text{WT_SKIP} \\
 \hline
 P; PA \vdash_{pc} \text{skip}
 \end{array}
 \quad
 \begin{array}{c}
 \text{WT_ASGN} \\
 \hline
 \frac{P(a) = \ell \quad pc \sqcup \ell \sqsubseteq P(x)}{P; PA \vdash_{pc} x := a}
 \end{array}
 \quad
 \begin{array}{c}
 \text{WT_SEQ} \\
 \hline
 \frac{P; PA \vdash_{pc} c_1 \quad P; PA \vdash_{pc} c_2}{P; PA \vdash_{pc} c_1; c_2}
 \end{array}
 \quad
 \begin{array}{c}
 \text{WT_IF} \\
 \hline
 \frac{P(be) = \ell \quad P; PA \vdash_{pc \sqcup \ell} c_1 \quad P; PA \vdash_{pc \sqcup \ell} c_2}{P; PA \vdash_{pc} \text{if } be \text{ then } c_1 \text{ else } c_2}
 \end{array}$$

$$\begin{array}{c}
 \text{WT_WHILE} \\
 \hline
 \frac{P(be) = \ell \quad P; PA \vdash_{pc \sqcup \ell} c}{P; PA \vdash_{pc} \text{while } be \text{ do } c}
 \end{array}
 \quad
 \begin{array}{c}
 \text{WT_AREAD} \\
 \hline
 \frac{P(i) = \ell_i \quad pc \sqcup \ell_i \sqcup PA(a) \sqsubseteq P(x)}{P; PA \vdash_{pc} x \leftarrow a[i]}
 \end{array}
 \quad
 \begin{array}{c}
 \text{WT_AWRITE} \\
 \hline
 \frac{P(i) = \ell_i \quad P(e) = \ell \quad pc \sqcup \ell_i \sqcup \ell \sqsubseteq PA(a)}{P; PA \vdash_{pc} a[i] \leftarrow e}
 \end{array}$$

IDEAL_IF

$$\frac{P(be) = \ell \quad b' = (\ell \vee \neg b) \wedge \llbracket be \rrbracket_\rho}{\langle \text{if } be \text{ then } c_{\mathbb{T}} \text{ else } c_{\mathbb{F}}, \rho, \mu, b \rangle \xrightarrow[\text{step}]{\text{branch } b'} \langle c_{b'}, \rho, \mu, b \rangle}$$

IDEAL_IF_FORCE

$$\frac{P(be) = \ell \quad b' = (\ell \vee \neg b) \wedge \llbracket be \rrbracket_\rho}{\langle \text{if } be \text{ then } c_{\mathbb{T}} \text{ else } c_{\mathbb{F}}, \rho, \mu, b \rangle \xrightarrow[\text{force}]{\text{branch } b'} \langle c_{\neg b'}, \rho, \mu, \mathbb{T} \rangle}$$

IDEAL_READ

$$\frac{P(ie) = \ell_i \quad i = \begin{cases} 0 & \text{if } (\neg \ell_i \vee P(X)) \wedge b \\ \llbracket ie \rrbracket_\rho & \text{otherwise} \end{cases} \quad v = \llbracket a[i] \rrbracket_\mu \quad i < |a|_\mu}{\langle X \leftarrow a[ie], \rho, \mu, b \rangle \xrightarrow[\text{step}]{\text{read } a^i} \langle \text{skip}, [X \mapsto v] \rho, \mu, b \rangle}$$

IDEAL_READ_FORCE

$$\frac{P(ie) \quad \neg P(X) \quad i = \llbracket ie \rrbracket_\rho \quad v = \llbracket b[j] \rrbracket_\mu \quad i \geq |a|_\mu \quad j < |b|_\mu}{\langle X \leftarrow a[ie], \rho, \mu, \mathbb{T} \rangle \xrightarrow[\text{load } b^j]{\text{read } a^i} \langle \text{skip}, [X \mapsto v] \rho, \mu, \mathbb{T} \rangle}$$

IDEAL_WRITE

$$\frac{i = \begin{cases} 0 & \text{if } (\neg \ell_i \vee \neg \ell) \wedge b \\ \llbracket ie \rrbracket_\rho & \text{otherwise} \end{cases} \quad P(ie) = \ell_i \quad P(ae) = \ell \quad v = \llbracket ae \rrbracket_\rho \quad i < |a|_\mu}{\langle a[ie] \leftarrow ae, \rho, \mu, b \rangle \xrightarrow[\text{step}]{\text{write } a^i} \langle \text{skip}, \rho, [a[i] \mapsto v] \mu, b \rangle}$$

IDEAL_WRITE_FORCE

$$\frac{P(ie) \quad P(ae) \quad i = \llbracket ie \rrbracket_\rho \quad v = \llbracket ae \rrbracket_\rho \quad i \geq |a|_\mu \quad j < |b|_\mu}{\langle a[ie] \leftarrow ae, \rho, \mu, \mathbb{T} \rangle \xrightarrow[\text{store } b^j]{\text{write } a^i} \langle \text{skip}, \rho, [b[j] \mapsto v] \mu, \mathbb{T} \rangle}$$

$$\begin{array}{c}
 \text{IDEAL_IF} \frac{P(\text{be}) \not\equiv \ell \quad b' = (\ell \vee \neg b) \wedge \llbracket \text{be} \rrbracket_\rho}{\langle \text{if } \text{be}_{@ \ell} \text{ then } \overline{c}_T \text{ else } \overline{c}_F, \rho, \mu, b, \text{pc}, P, PA \rangle \xrightarrow[\text{step}]{\text{branch } b'} \langle \text{branch } \text{pc } \overline{c}_{b'}, \rho, \mu, b, \text{pc} \sqcup \ell, P, PA \rangle} \\
 \\
 \text{IDEAL_IF_FORCE} \frac{P(\text{be}) \not\equiv \ell \quad b' = (\ell \vee \neg b) \wedge \llbracket \text{be} \rrbracket_\rho}{\langle \text{if } \text{be}_{@ \ell} \text{ then } \overline{c}_T \text{ else } \overline{c}_F, \rho, \mu, b, \text{pc}, P, PA \rangle \xrightarrow[\text{force}]{\text{branch } b'} \langle \text{branch } \text{pc } \overline{c}_{b'}, \rho, \mu, T, \text{pc} \sqcup \ell, P, PA \rangle} \\
 \\
 \text{IDEAL_READ} \frac{P(\text{ie}) \not\equiv \ell_i \quad i = \begin{cases} 0 & \text{if } \neg \ell_i \wedge b \\ \llbracket \text{ie} \rrbracket_\rho & \text{otherwise} \end{cases} \quad v = \begin{cases} 0 & \text{if } \ell_x \wedge \ell_i \wedge b \\ \llbracket a[i] \rrbracket_\mu & \text{otherwise} \end{cases} \quad i < |a|_\mu}{\langle X_{@ \ell_x} \leftarrow a[\text{ie}_{@ \ell_i}], \rho, \mu, b, \text{pc}, P, PA \rangle \xrightarrow[\text{step}]{\text{read } a[i]} \langle \text{skip}, [X \mapsto v] \rho, \mu, b, \text{pc}, [X \mapsto \ell_x] P, PA \rangle} \\
 \\
 \text{IDEAL_READ_FORCE} \frac{P(\text{ie}) \not\equiv \ell_i \quad i = \llbracket \text{ie} \rrbracket_\rho \quad v = \begin{cases} 0 & \text{if } \ell_x \\ \llbracket b[j] \rrbracket_\mu & \text{otherwise} \end{cases} \quad i \geq |a|_\mu \quad j < |b|_\mu}{\langle X_{@ \ell_x} \leftarrow a[\text{ie}_{@ T}], \rho, \mu, T, \text{pc}, P, PA \rangle \xrightarrow[\text{step}]{\text{read } a[i]} \langle \text{skip}, [X \mapsto v] \rho, \mu, T, \text{pc}, [X \mapsto \ell_x] P, PA \rangle} \\
 \\
 \text{IDEAL_SEQ_SKIP} \frac{\text{terminal } \overline{c}_1}{\langle \overline{c}_1; @ (P', PA') \overline{c}_2, \rho, \mu, b, \text{pc}, P, PA \rangle \xrightarrow{\bullet} \langle \overline{c}_2, \rho, \mu, b, \text{pc-after } \overline{c}_1 \text{ pc}, P, PA \rangle} \\
 \\
 \text{IDEAL_BRANCH} \frac{\langle \overline{c}, \rho, \mu, b, \text{pc}, P, PA \rangle \xrightarrow[d]{o} \langle \overline{c}', \rho', \mu', b', \text{pc}', P', PA' \rangle}{\langle \text{branch } \ell \overline{c}, \dots \rangle \xrightarrow[d]{o} \langle \text{branch } \ell \overline{c}', \dots \rangle}
 \end{array}$$

Fig. 12: Ideal semantics for FvSLH[∀] (selected rules)

$$\begin{aligned}
 \llbracket \text{skip} \rrbracket_{pc}^{P,PA} &\doteq (\text{skip}, P, PA) \\
 \llbracket X := e \rrbracket_{pc}^{P,PA} &\doteq (X := e, [X \mapsto P(e)]P, PA) \\
 \llbracket c_1; c_2 \rrbracket_{pc}^{P,PA} &\doteq (\overline{c_1}; @_{(P_1, PA_1)} \overline{c_2}, P_2, PA_2) \text{ where } (\overline{c_1}, P_1, PA_1) = \llbracket c_1 \rrbracket_{pc}^{P,PA} \\
 &\quad \text{and } (\overline{c_2}, P_2, PA_2) = \llbracket c_2 \rrbracket_{pc}^{P_1, PA_1} \\
 \llbracket \text{if } be \text{ then } c_1 \text{ else } c_2 \rrbracket_{pc}^{P,PA} &\doteq (\text{if } be_{@P(be)} \text{ then } \overline{c_1} \text{ else } \overline{c_2}, P_1 \sqcup P_2, PA_1 \sqcup PA_2) \text{ where } (\overline{c_1}, P_1, PA_1) = \llbracket c_1 \rrbracket_{pc \sqcup P(be)}^{P,PA} \\
 &\quad \text{and } (\overline{c_2}, P_2, PA_2) = \llbracket c_2 \rrbracket_{pc \sqcup P(be)}^{P,PA} \\
 \llbracket \text{while } be \text{ do } c \rrbracket_{pc}^{P,PA} &\doteq (\text{while } be_{@P_{fix}(be)} \text{ do } \overline{c}_{@ (P_{fix}, PA_{fix})}, P_{fix}, PA_{fix}) \\
 &\quad \text{where } (P_{fix}, PA_{fix}) = \mathbf{fix} \ (\lambda(P', PA'). \mathbf{let} \ (\overline{c}, P'', PA'') = \llbracket c \rrbracket_{pc \sqcup P'(be)}^{P', PA'} \mathbf{in} \ (P'', PA'') \sqcup (P, PA)) \\
 \llbracket X \leftarrow a[i] \rrbracket_{pc}^{P,PA} &\doteq (X_{@pc \sqcup P(i) \sqcup PA(a)} \leftarrow a[i_{@P(i)}], [X \mapsto pc \sqcup P(i) \sqcup PA(a)]P, PA) \\
 \llbracket a[i] \leftarrow e \rrbracket_{pc}^{P,PA} &\doteq (a[i_{@P(i)}] \leftarrow e, P, [a \mapsto PA(a) \sqcup pc \sqcup P(i) \sqcup P(e)]PA)
 \end{aligned}$$

Fig. 11: Flow-sensitive IFC analysis generating annotated commands

$$\text{WL_SKIP} \frac{(P_1, PA_1) \sqsubseteq (P_2, PA_2)}{P_1, PA_1 \rightsquigarrow P_2, PA_2 \vdash_{pc} \text{skip}}$$

$$\text{WL_ASGN} \frac{([X \mapsto P_1(e)]P_1, PA_1) \sqsubseteq P_2, PA_2}{P_1, PA_1 \rightsquigarrow P_2, PA_2 \vdash_{pc} (X := e)}$$

$$\text{WL_SEQ} \frac{\begin{array}{l} \text{branch-free } \bar{c}_2 \quad P_1, PA_1 \rightsquigarrow P', PA' \vdash_{pc} \bar{c}_1 \\ P', PA' \rightsquigarrow P_2, PA_2 \vdash_{(pc\text{-after } \bar{c}_1 \text{ } pc)} \bar{c}_2 \end{array}}{P_1, PA_1 \rightsquigarrow P_2, PA_2 \vdash_{pc} (\bar{c}_1 ; @ (P', PA') \bar{c}_2)}$$

$$\text{WL_IF} \frac{\begin{array}{l} P_1(be) \sqsubseteq \ell_{be} \quad \text{branch-free } \bar{c}_1 \quad \text{branch-free } \bar{c}_2 \\ P_1, PA_1 \rightsquigarrow P_2, PA_2 \vdash_{pc \sqcup \ell_{be}} \bar{c}_1 \quad P_1, PA_1 \rightsquigarrow P_2, PA_2 \vdash_{pc \sqcup \ell_{be}} \bar{c}_2 \end{array}}{P_1, PA_1 \rightsquigarrow P_2, PA_2 \vdash_{pc} (\text{if } be_{@ \ell_{be}} \text{ then } \bar{c}_1 \text{ else } \bar{c}_2)}$$

WL_WHILE

$$\frac{\begin{array}{l} P_1(be) \sqsubseteq \ell_{be} \quad \text{branch-free } \bar{c} \\ (P_1, PA_1) \sqsubseteq (P', PA') \quad (P', PA') \sqsubseteq (P_2, PA_2) \\ P', PA' \rightsquigarrow P', PA' \vdash_{pc \sqcup \ell_{be}} \bar{c}_1 \end{array}}{P_1, PA_1 \rightsquigarrow P_2, PA_2 \vdash_{pc} (\text{while } be_{@ \ell_{be}} \text{ do } \bar{c}_{@ (P', PA')})}$$

WL_AREAD

$$\frac{\begin{array}{l} P_1(e) \sqsubseteq \ell_i \quad pc \sqsubseteq \ell_x \\ \ell_i \sqsubseteq \ell_x \quad PA_1(a) \sqsubseteq \ell_x \quad ([X \mapsto \ell_x]P_1, PA_1) \sqsubseteq (P_2, PA_2) \end{array}}{P_1, PA_1 \rightsquigarrow P_2, PA_2 \vdash_{pc} (X_{@ \ell_x} \leftarrow a[e_{@ \ell_i}])}$$

WL_AWRITE

$$\frac{\begin{array}{l} P_1(i) \sqsubseteq \ell_i \\ (P_1, [a \mapsto PA_1(a) \sqcup pc \sqcup \ell_i \sqcup P_1(e)]PA_1) \sqsubseteq (P_2, PA_2) \end{array}}{P_1, PA_1 \rightsquigarrow P_2, PA_2 \vdash_{pc} (a[i_{@ \ell_i}] \leftarrow e)}$$

$$\text{WL_BRANCH} \frac{P_1, PA_1 \rightsquigarrow P_2, PA_2 \vdash_{pc} \bar{c}}{P_1, PA_1 \rightsquigarrow P_2, PA_2 \vdash_{pc} (\text{branch } \ell \bar{c})}$$

```
if i < secrets_size then  
  secrets[i] <- key;  
  x <- a[0];  
  if x then...
```

- out-of-bounds `i` could write to `a[0]`
- read from public array `a` is unprotected
 - reads speculatively stored secret