Hybrid Monitoring of Attacker Knowledge

Frédéric Besson, Nataliia Bielova, Thomas Jensen
INRIA

IEEE Computer Security Foundations 2016
June 29, 2016

Information flow control

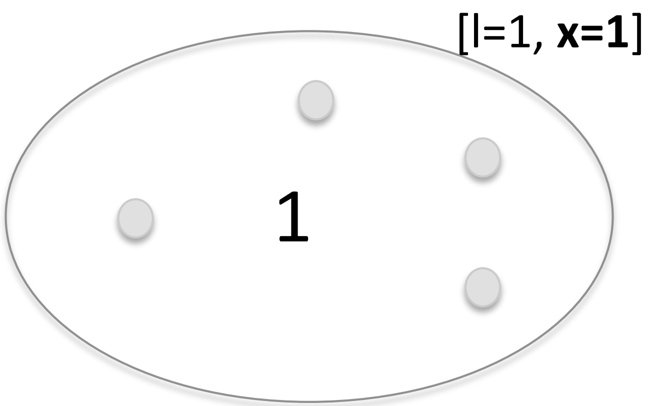
Noninterference	Quantitative Information Flow
Secret input does not flow into public output	How much information (in bits) about secret a program leaks to the output?

Information flow control

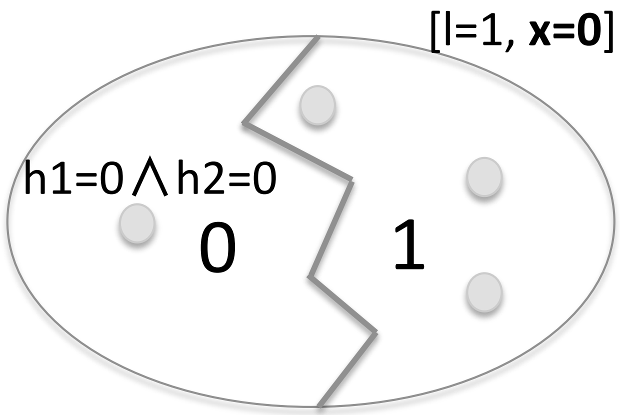
Noninterference	Attacker's knowledge	Quantitative Information Flow
Secret input does not flow into public output	What information about the secret is flown to the output in concrete program execution?	How much information (in bits) about secret a program leaks to the output?

A program which is not secure

```
if h1 = 1 then x = 1
else skip;
if h2 = 1 then l = 1
else l = x;
output l
```



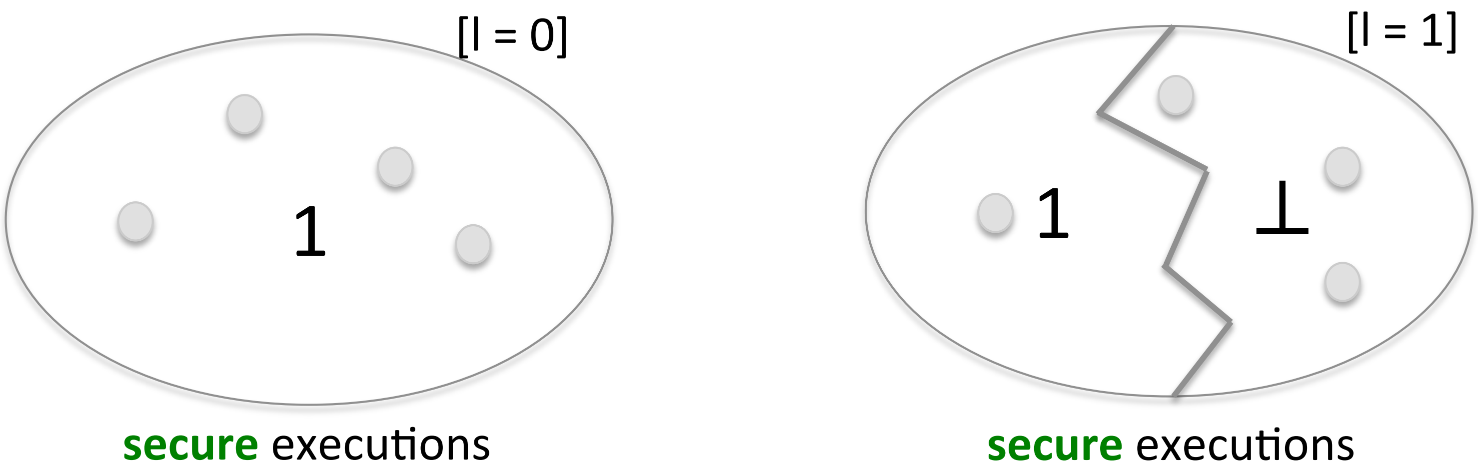
set of **secure** executions
starting with l=1, x=1



set of **insecure** executions
starting with l=1, x=0

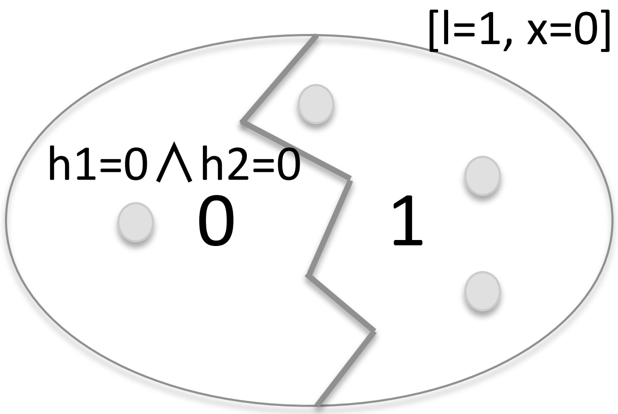
Security Definition

- **TINI: Termination-Insensitive Noninterference**
  - Program P is TINI if for all low-equivalence classes:



What does an attacker learn?

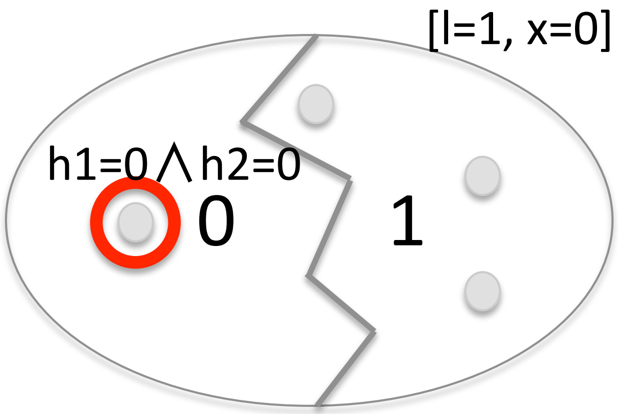
```
if h1 = 1 then x = 1
else skip;
if h2 = 1 then l = 1
else l = x;
output l
```



insecure executions

What does an attacker learn?

```
if h1 = 1 then x = 1
else skip;
if h2 = 1 then l = 1
else l = x;
output l
```



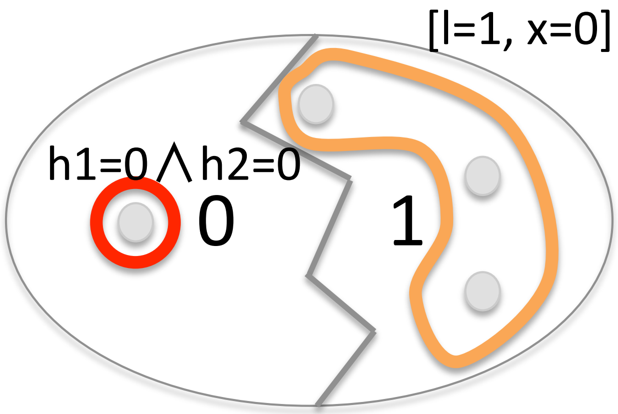
h1=0 ∧ h2=0

attacker knows
values of both secrets

insecure executions

What does an attacker learn?

```
if h1 = 1 then x = 1
else skip;
if h2 = 1 then l = 1
else l = x;
output l
```



h1=0 ∧ h2=0

attacker knows
values of both secrets

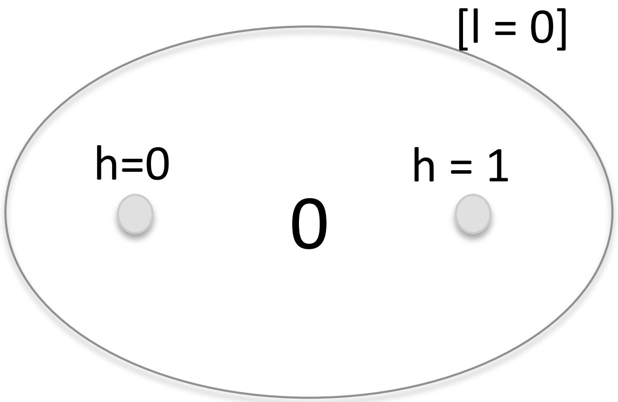
insecure executions

h1=1 ∨ h2=1

attacker knows
some information about secrets

A program which is secure

```
l = 0;
if h = 1 then skip
else
    x = 5;
    while x > 0 do
        x = x-1; l = x;
output l
```



secure executions

- **Does any dynamic/hybrid monitor accept all executions of this program?**

A program which is secure

```
l = 0;
if h = 1 then skip
else
    x = 5;
    while x > 0 do
        x = x-1; l = x;
output l
```

Dynamic [h=0]	Hybrid [h=1]
… branch taken block execution due to low assignment in high context	… branch taken static analysis … block execution since l could be modified in else-branch

- **Dynamic monitors block too early**
  - [Zdancewic ’02, Austin and Flanagan ’10]
- **Hybrid monitors block due to imprecision of static analysis**
  - [Le Guernic ‘07, Russo and Sabelfeld ’10, Besson et al ‘13]

Challenges

- **How to track attacker's knowledge?**

- **How to make a monitor accept more secure executions?**

Answer:
Hybrid monitoring of attacker's knowledge

Hybrid monitor

Dynamic + Static analysis

- Dynamic analysis monitors one execution
- Static analysis is called on-the-fly for non-executed branches

- Two sets of rules: one for dynamic + one for static

Hybrid monitor

$$\boxed{(\mathrm{P}, \rho, \kappa) \Downarrow (\rho', \kappa')}$$

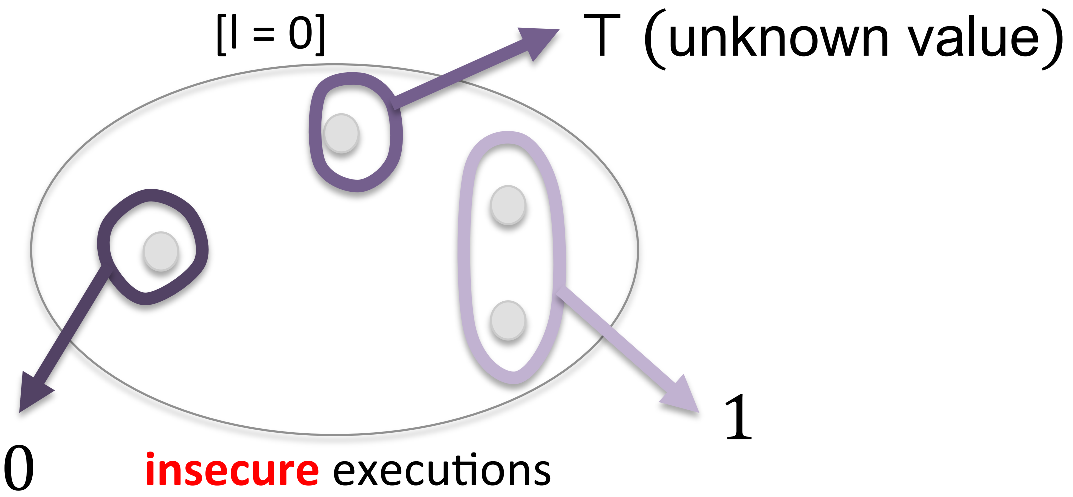
- ρ, ρ': $\mathrm{Env} \cup \{\cdot\}$
- κ, κ' : $\mathrm{Var} \to \mathbf{K}$ labeling with knowledge
- Env for dynamic analysis
- $\cdot$ for static analysis

Expressive knowledge domain

$$\boxed{\kappa\text{: Var} \rightarrow \mathbf{K}}$$

- $\kappa(\mathrm{x})$ splits the initial environments in equivalence classes w.r.t. the possible values of x

$\kappa(\mathrm{x})$:



Expressive knowledge domain

$$\boxed{\kappa\colon \mathrm{Var} \to \mathbf{K}}$$

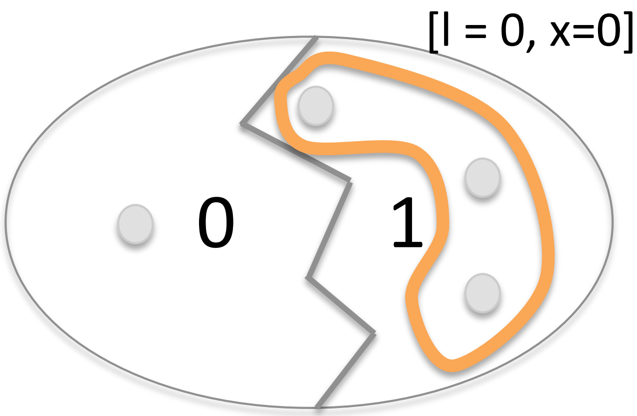
$$\boxed{\mathbf{K} \triangleq \mathrm{Env} \to \mathrm{Value} \cup \{\top, \bot\}}$$

- $\kappa(\mathrm{x})(\rho) = v$ if the program terminates then x has value v
- $\kappa(\mathrm{x})(\rho) = \top$ no information (x can have any value)
- $\kappa(\mathrm{x})(\rho) = \bot$ the program certainly does not terminate on ρ

[h1 = 0, h2 = 1]

```
if h1 = 1 then x = 1
else skip;
if h2 = 1 then l = 1
else l = x;
output l
```

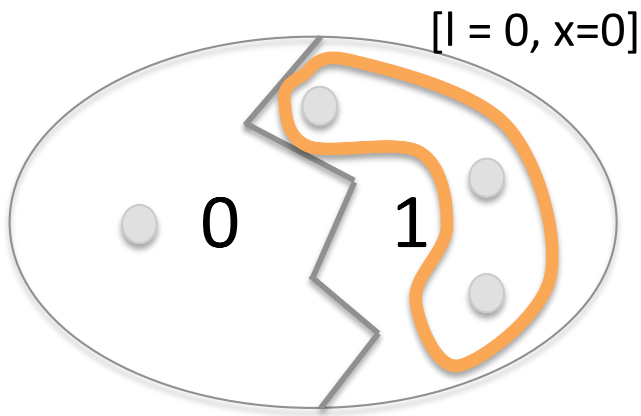
REAL KNOWLEDGE



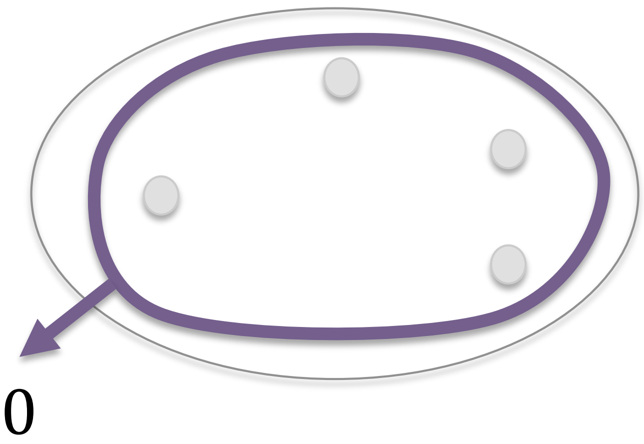
[h1 = 0, h2 = 1]

```
if h1 = 1 then x = 1
else skip;
if h2 = 1 then l = 1
else l = x;
output l
```

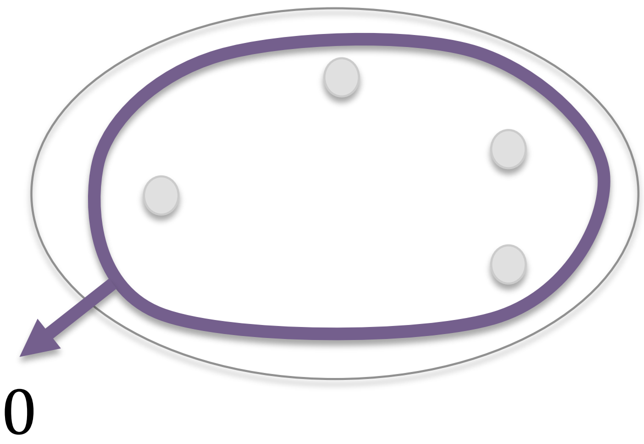
REAL KNOWLEDGE



$\kappa(\mathtt{x})$:



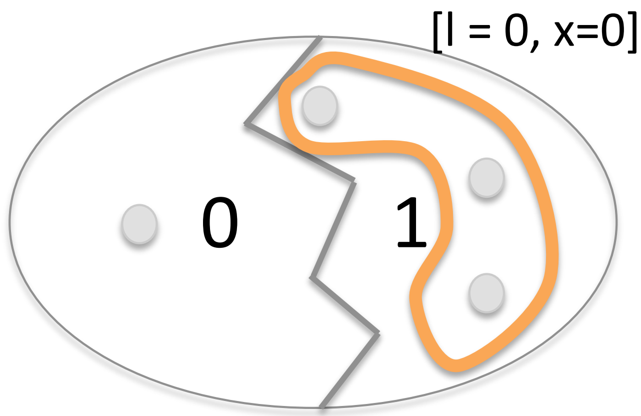
$\kappa(\mathtt{l})$:



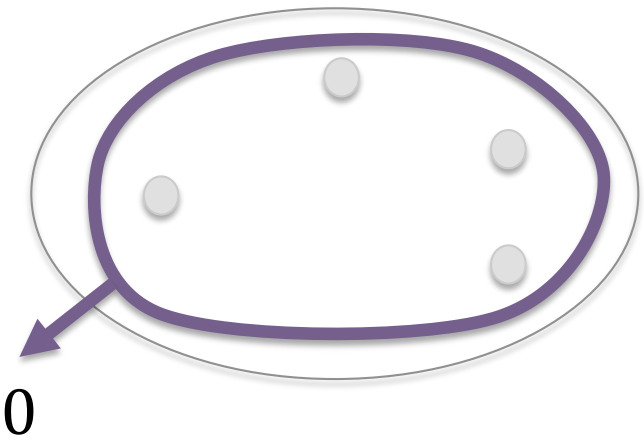
[h1 = 0, h2 = 1]

```
if h1 = 1 then x = 1
else skip;
if h2 = 1 then l = 1
else l = x;
output l
```

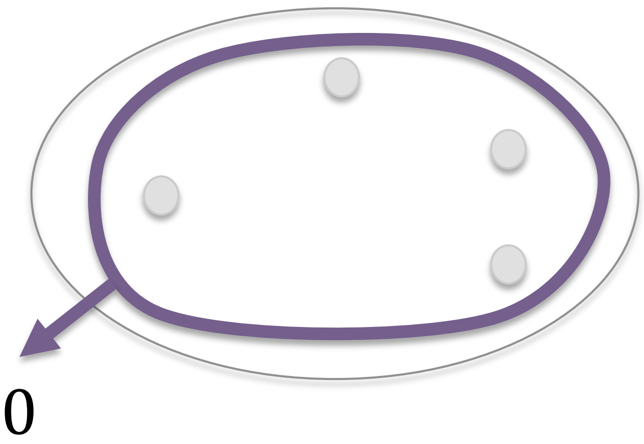
REAL KNOWLEDGE



$\kappa(\mathtt{x})$:



$\kappa(\mathtt{l})$:

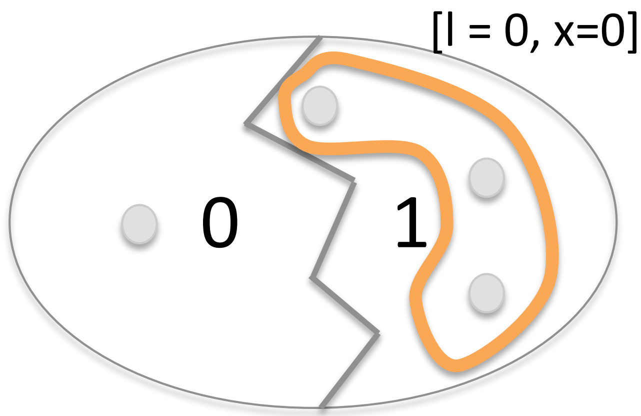


[h1 = 0, h2 = 1]

static analysis

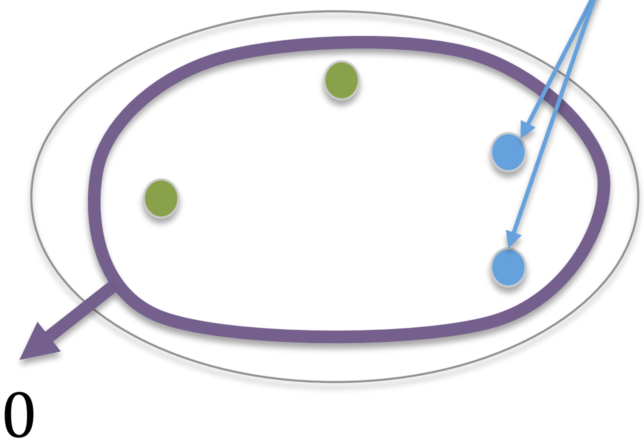
```
if h1 = 1 then x = 1
else skip;
if h2 = 1 then l = 1
else l = x;
output l
```

REAL KNOWLEDGE

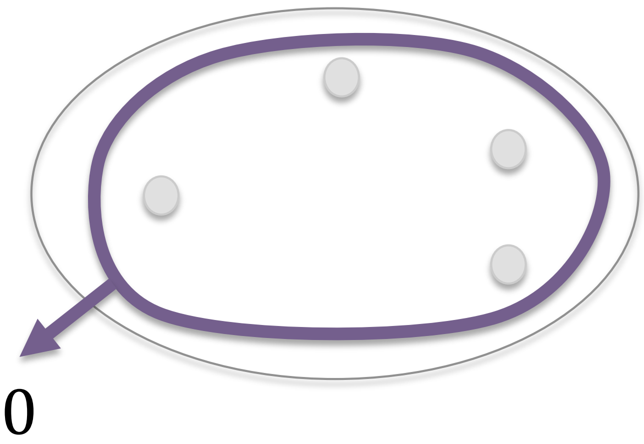


The result of static analysis only applies to environments where h1=1

$\kappa(\mathtt{x})$:



$\kappa(\mathtt{l})$:

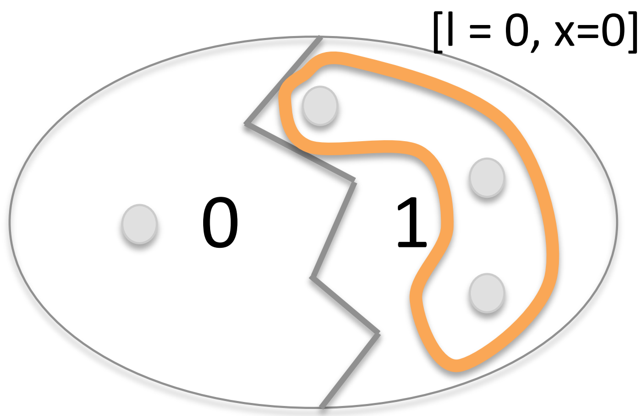


[h1 = 0, h2 = 1]

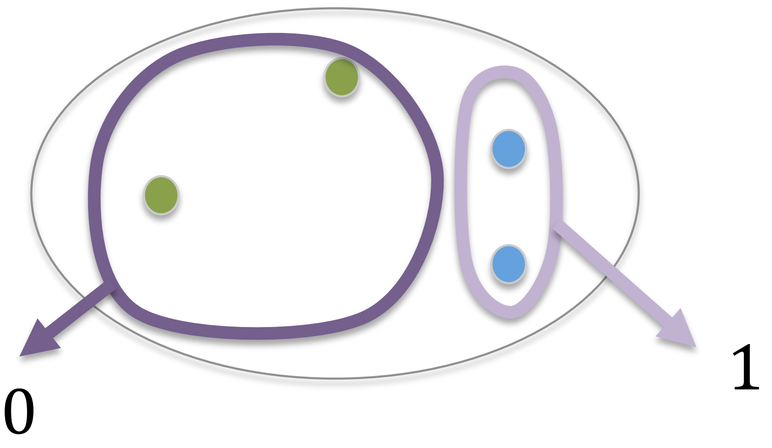
static analysis

```
if h1 = 1 then x = 1
else skip;
if h2 = 1 then l = 1
else l = x;
output l
```

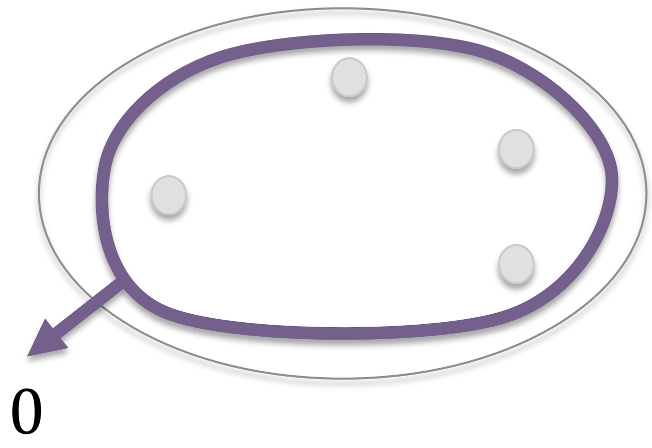
REAL KNOWLEDGE



$\kappa(\mathtt{x})$:



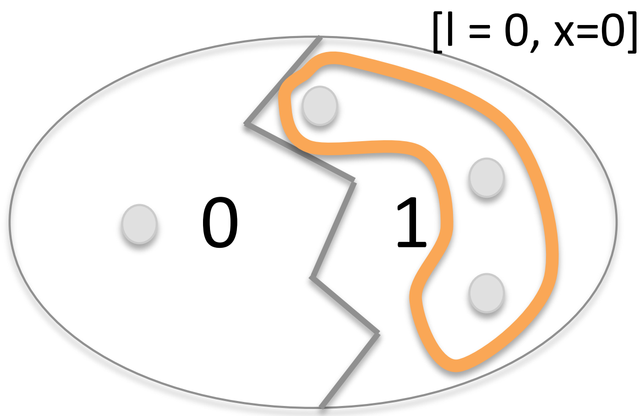
$\kappa(\mathtt{l})$:

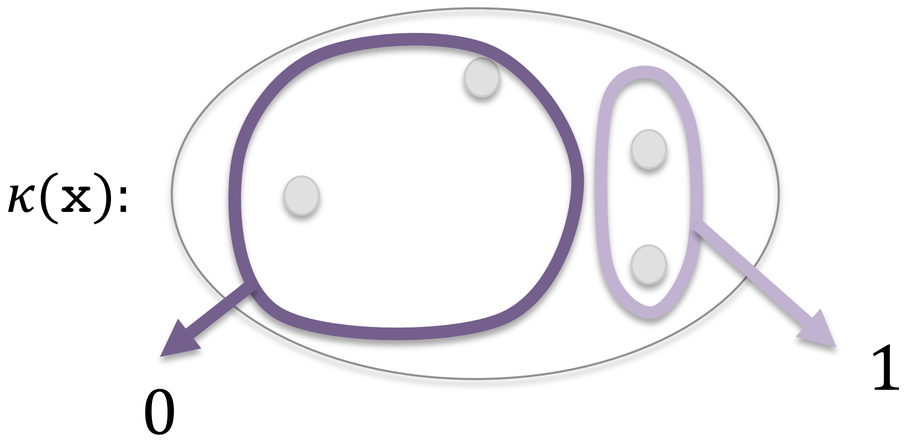


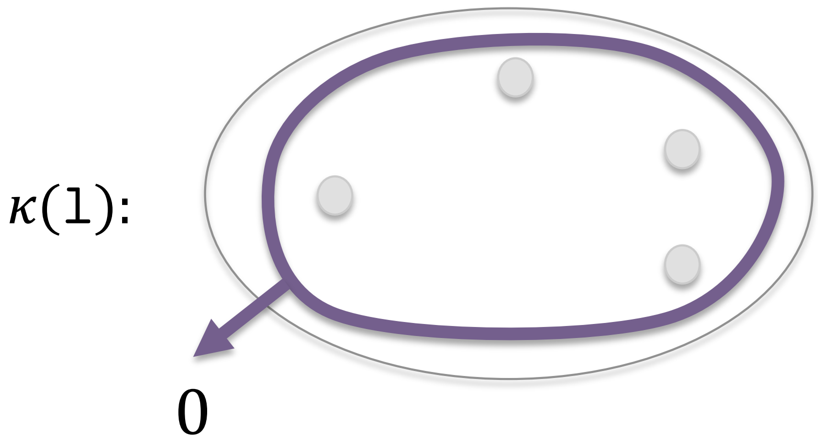
[h1 = 0, h2 = 1]

```
if h1 = 1 then x = 1
else skip;
if h2 = 1 then l = 1
else l = x;
output l
```

REAL KNOWLEDGE





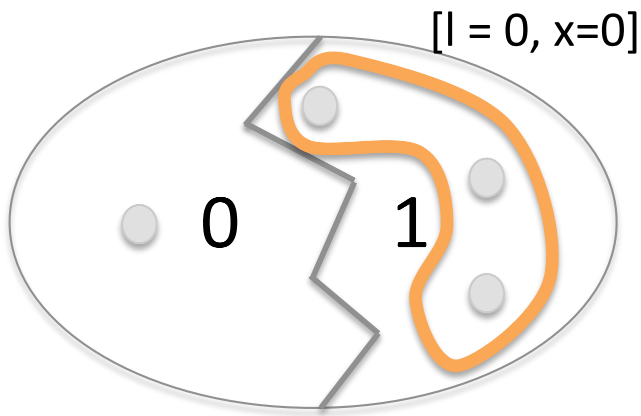


[h1 = 0, h2 = 1]

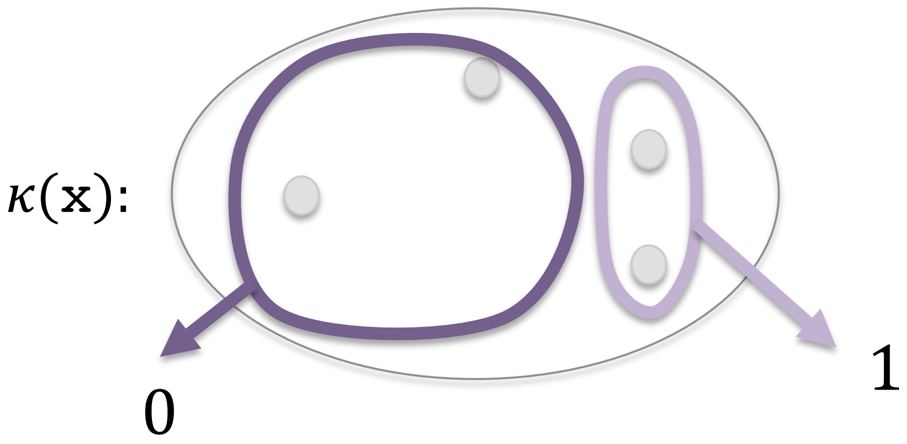
```
if h1 = 1 then x = 1
else skip;
if h2 = 1 then l = 1
else l = x;
output l
```

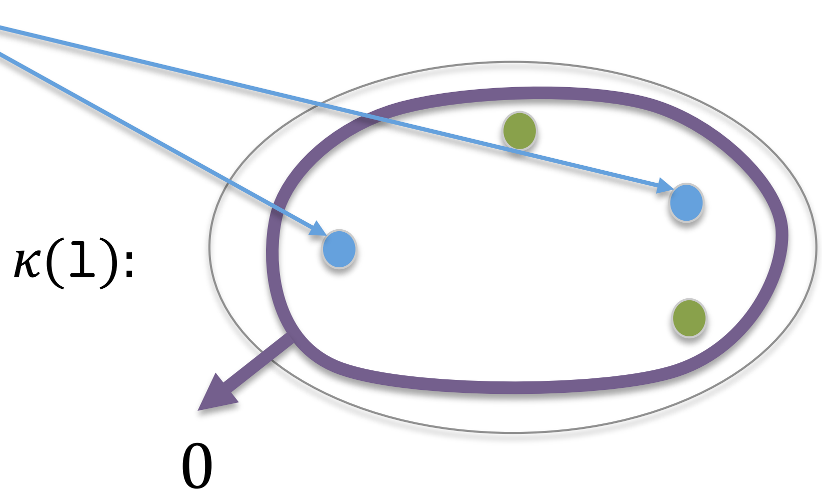
static analysis

REAL KNOWLEDGE



The result of static analysis only applies to environments where h2=0



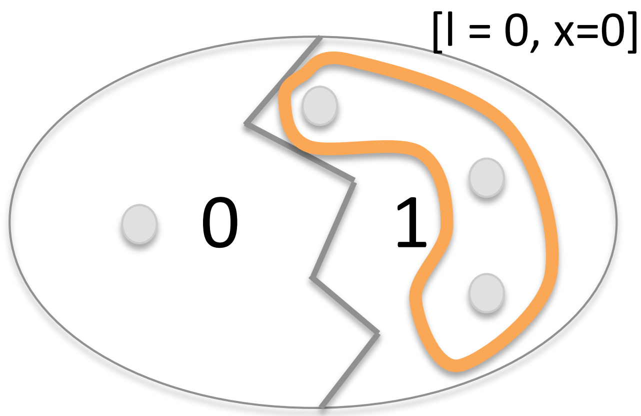


[h1 = 0, h2 = 1]

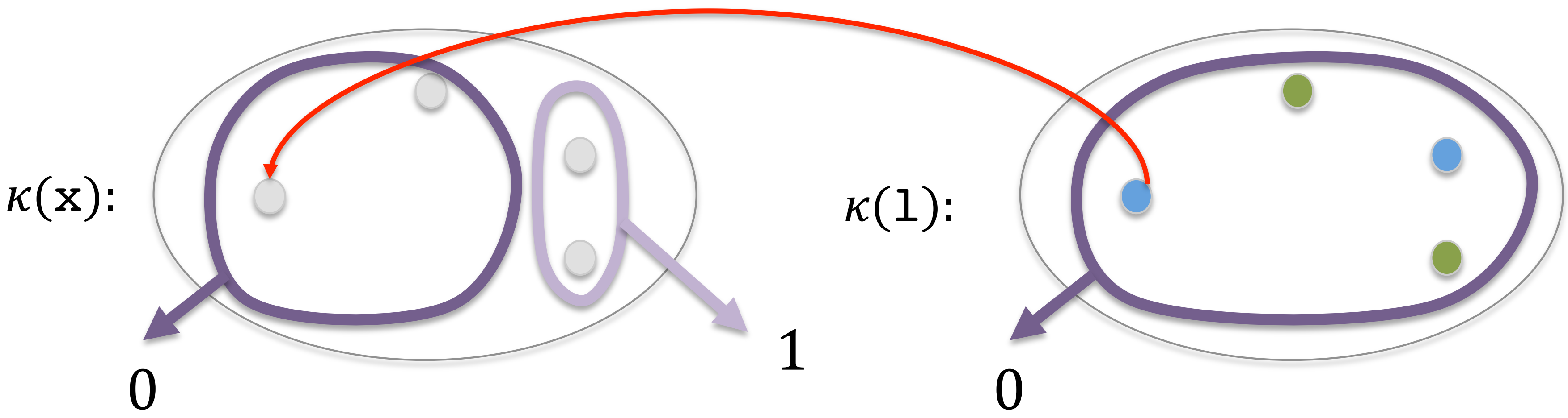
```
if h1 = 1 then x = 1
else skip;
if h2 = 1 then l = 1
else l = x;
output l
```

static analysis

REAL KNOWLEDGE



The new knowledge in these environments comes from the knowledge of x

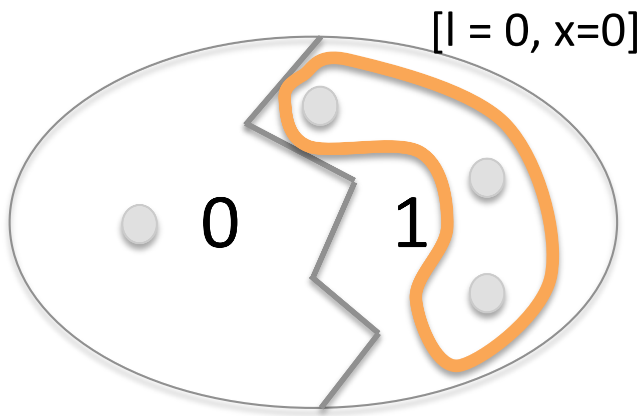


[h1 = 0, h2 = 1]

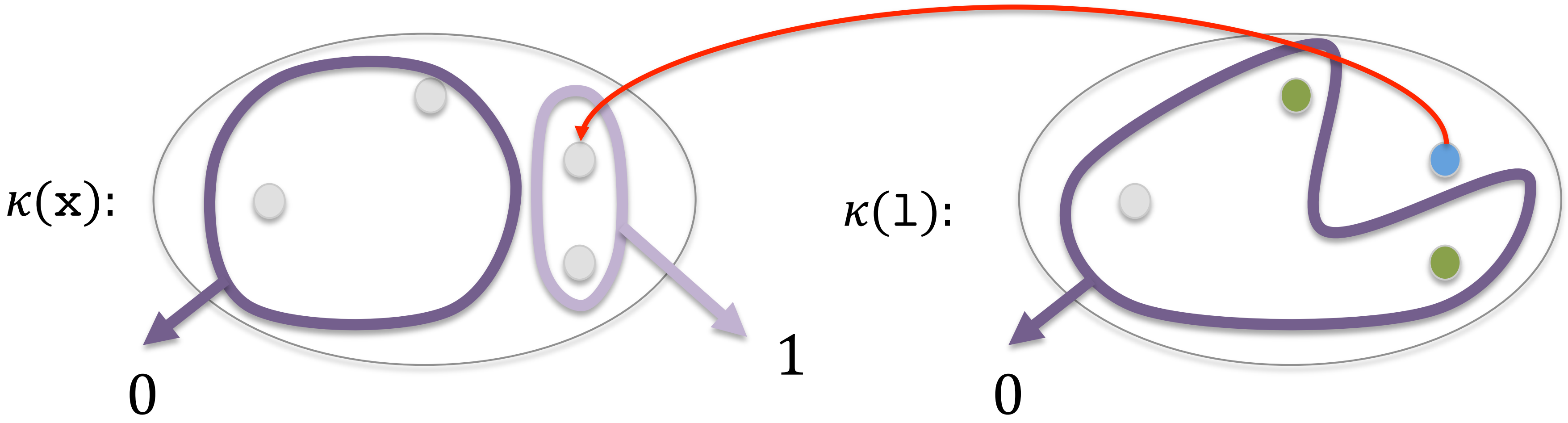
```
if h1 = 1 then x = 1
else skip;
if h2 = 1 then l = 1
else l = x;
output l
```

static analysis

REAL KNOWLEDGE



The new knowledge in these environments comes from the knowledge of x

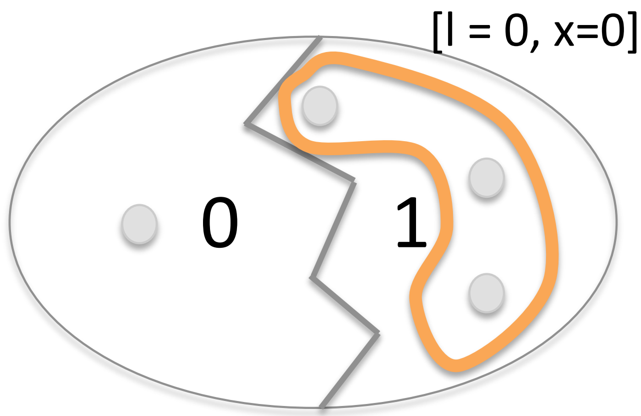


[h1 = 0, h2 = 1]

```
if h1 = 1 then x = 1
else skip;
if h2 = 1 then l = 1
else l = x;
output l
```

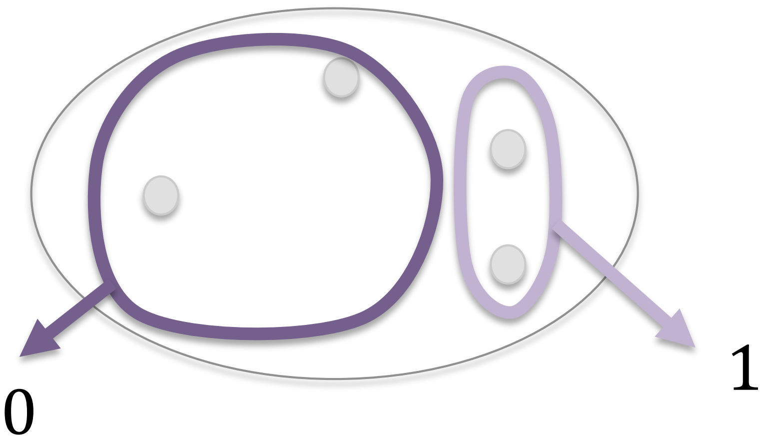
static analysis

REAL KNOWLEDGE

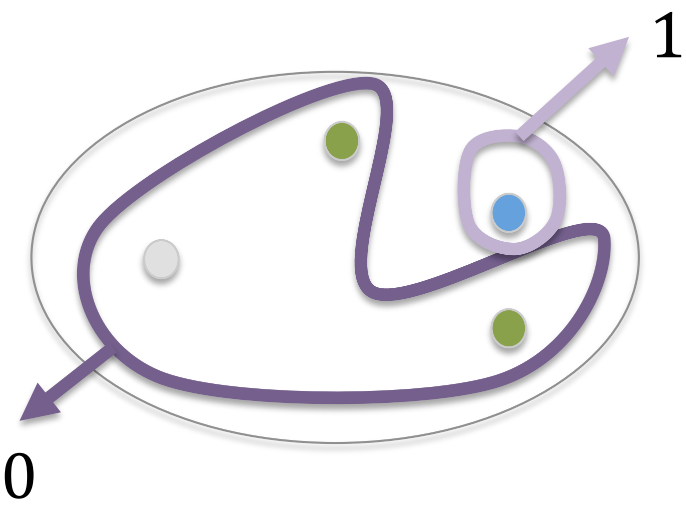


The new knowledge in these environments comes from the knowledge of x

$\kappa(\mathtt{x})$:



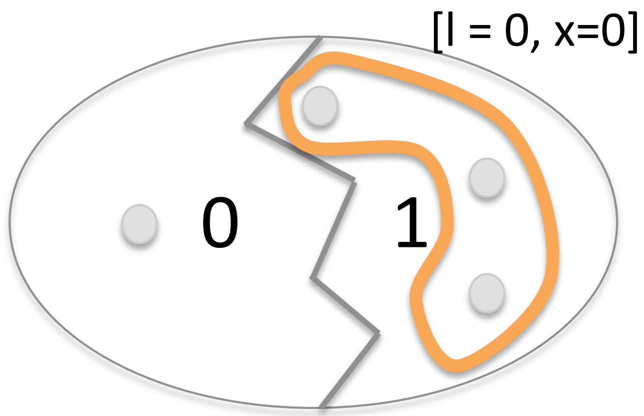
$\kappa(\mathtt{l})$:



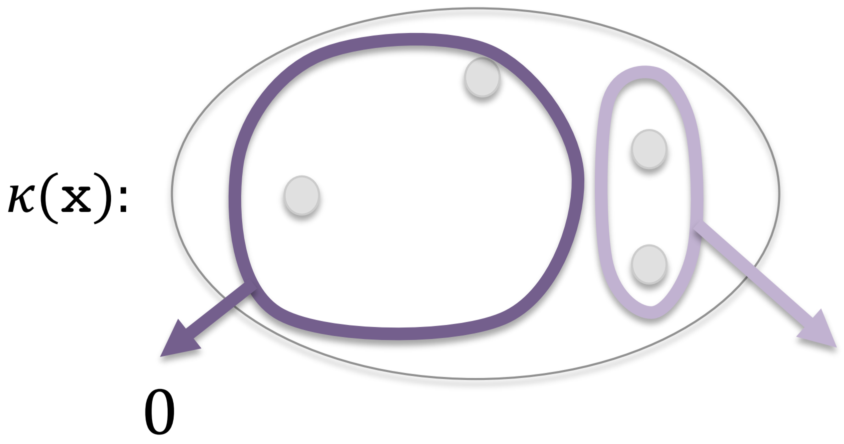
[h1 = 0, h2 = 1]

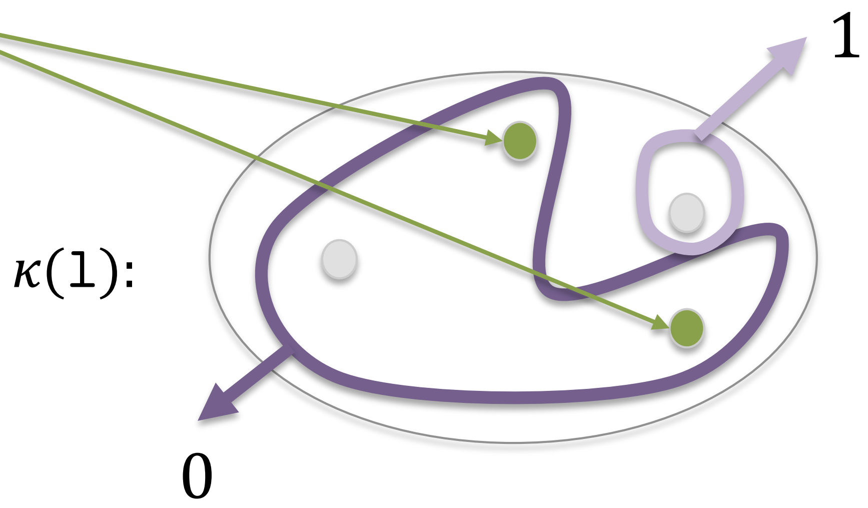
```
if h1 = 1 then x = 1
else skip;
if h2 = 1 then l = 1
else l = x;
output l
```

REAL KNOWLEDGE



The knowledge in current execution applies to environments where h2=1

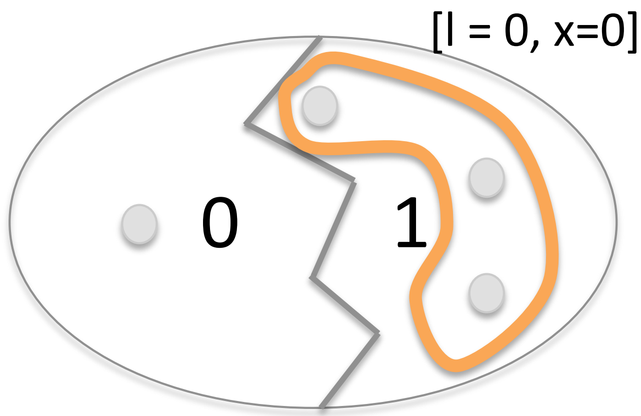




[h1 = 0, h2 = 1]

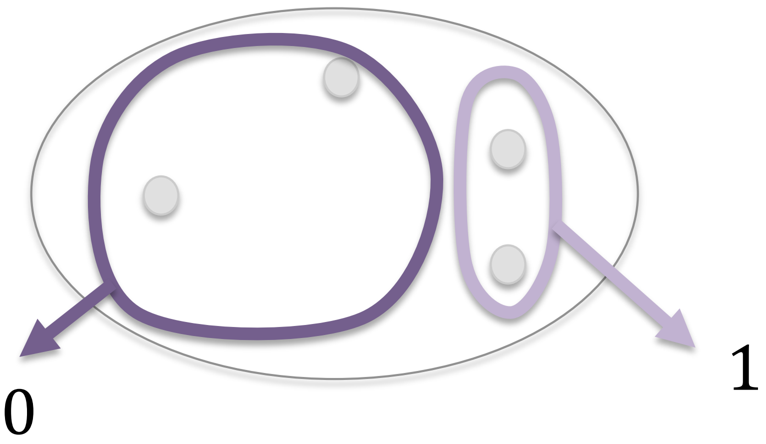
```
if h1 = 1 then x = 1
else skip;
if h2 = 1 then l = 1
else l = x;
output l
```

REAL KNOWLEDGE

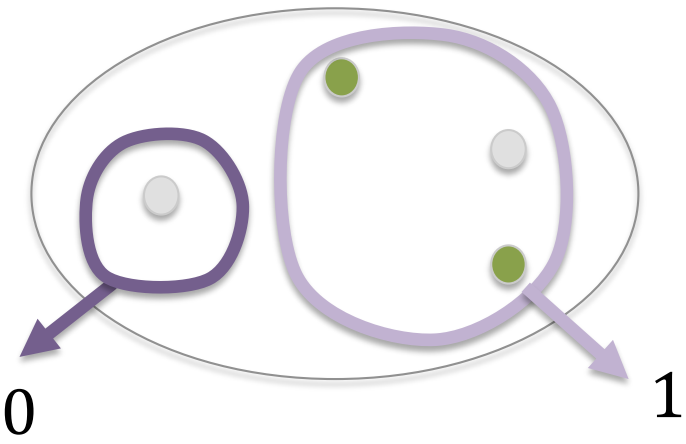


The knowledge in current execution applies to environments where h2=1

$\kappa(\mathtt{x})$:



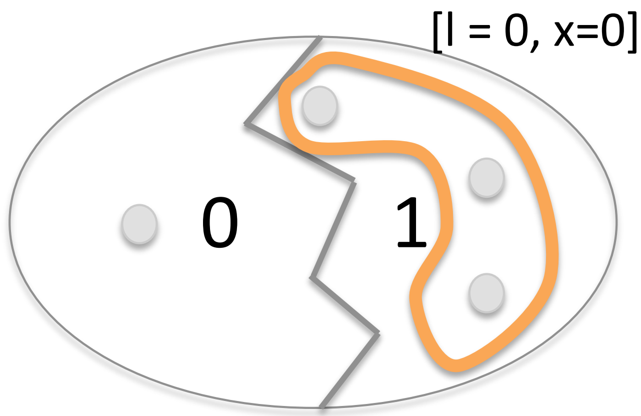
$\kappa(\mathtt{l})$:



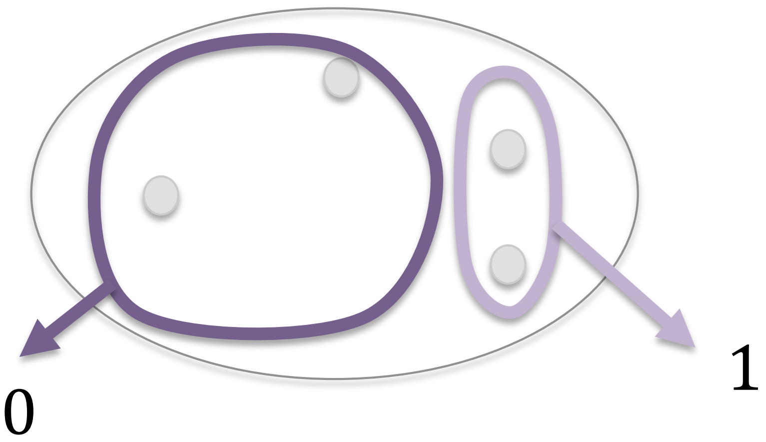
[h1 = 0, h2 = 1]

```
if h1 = 1 then x = 1
else skip;
if h2 = 1 then l = 1
else l = x;
output l
```

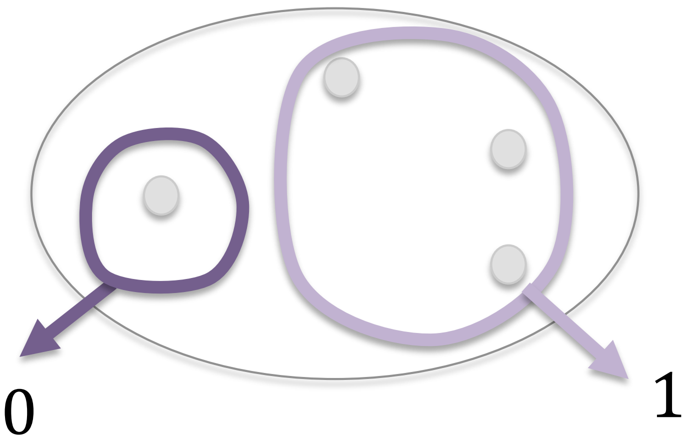
REAL KNOWLEDGE



$\kappa(\mathtt{x})$:



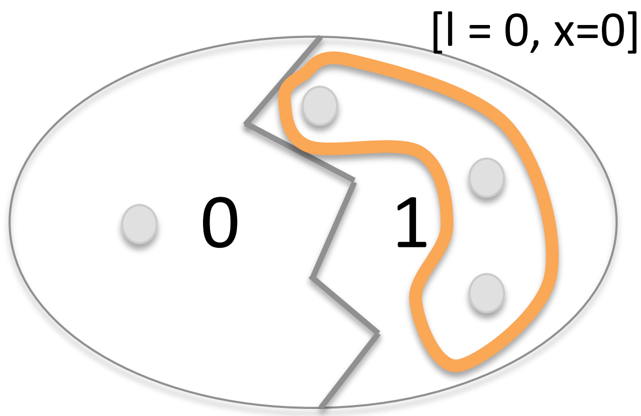
$\kappa(\mathtt{l})$:

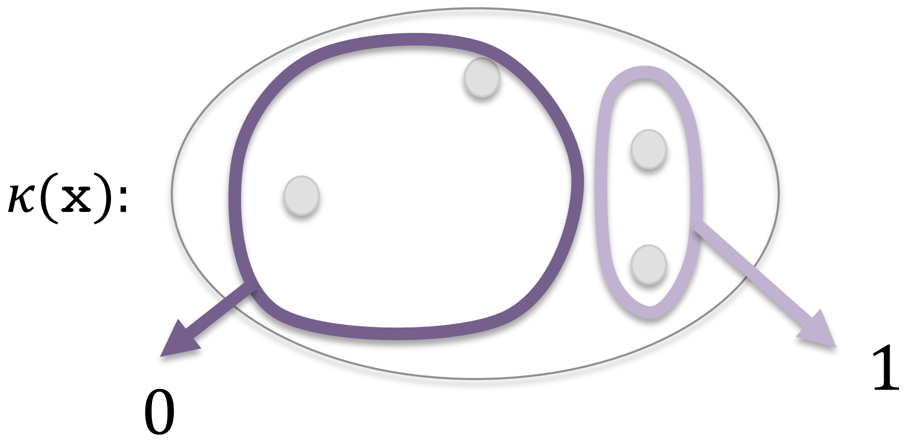


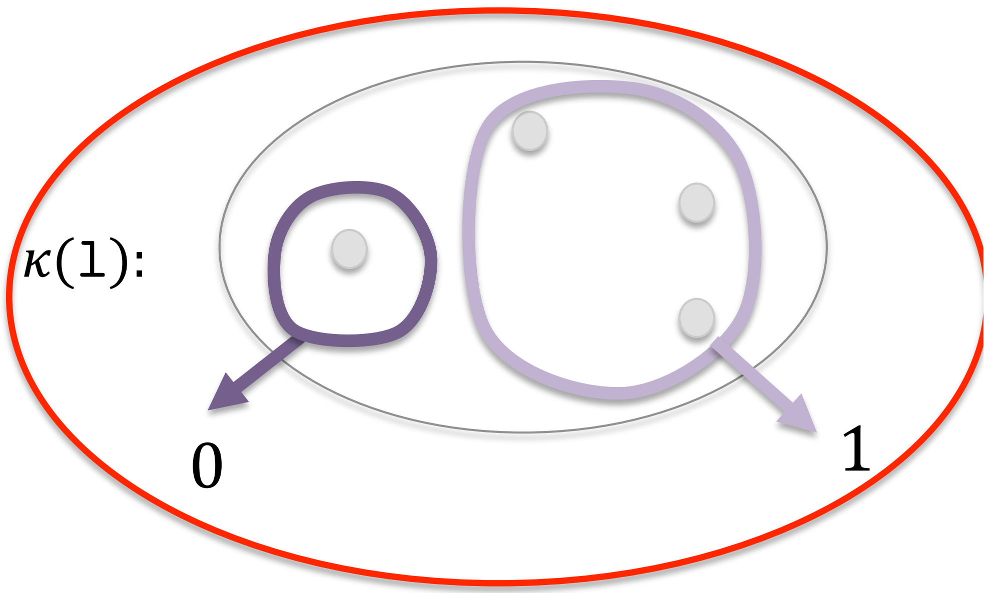
[h1 = 0, h2 = 1]

```
if h1 = 1 then x = 1
else skip;
if h2 = 1 then l = 1
else l = x;
output l
```

REAL KNOWLEDGE



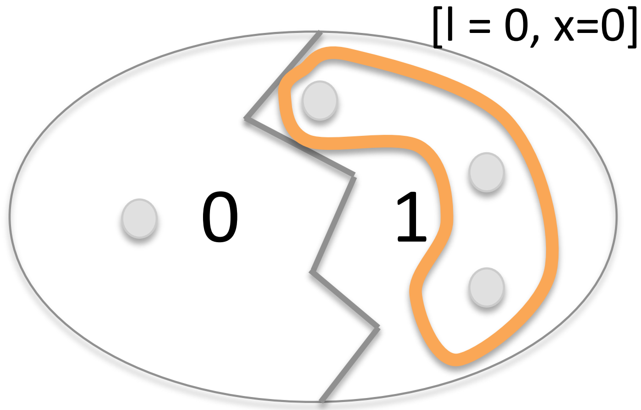




[h1 = 0, h2 = 1]

```
if h1 = 1 then x = 1
else skip;
if h2 = 1 then l = 1
else l = x;
output l
```

REAL KNOWLEDGE

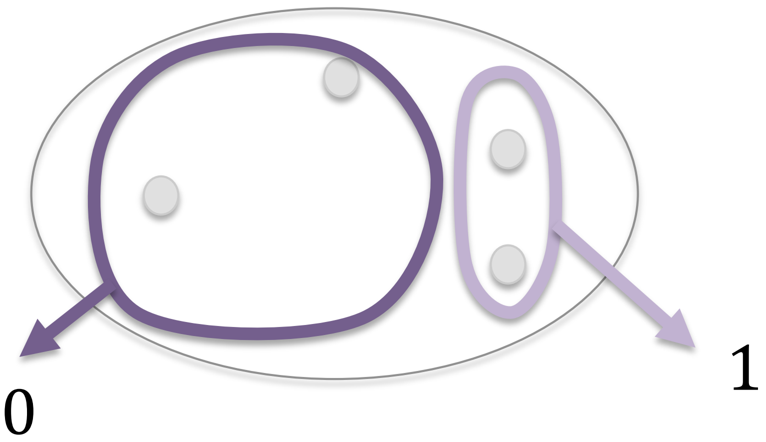


REAL KNOWLEDGE

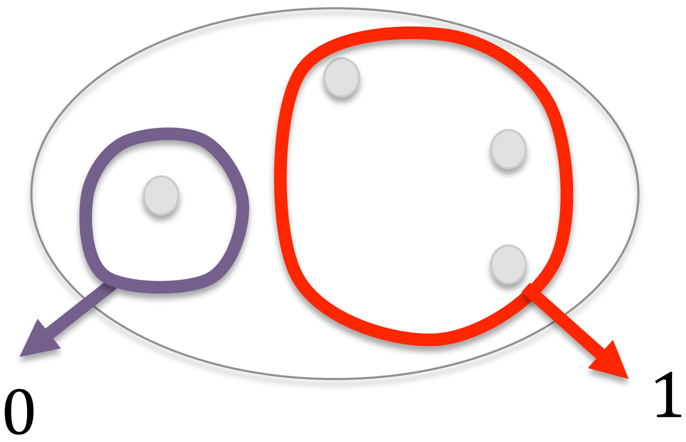
=

APPROXIMATED KNOWLEDGE

$\kappa(\mathtt{x})$:



$\kappa(\mathtt{l})$:



Implementation

- Symbolic representation of knowledge

$$\boxed{\mathbf{K}^\flat \subset \mathcal{P}(\mathbb{F}\times\mathbb{E}) \times \mathbb{F}}$$

propositional formulas **program expressions**

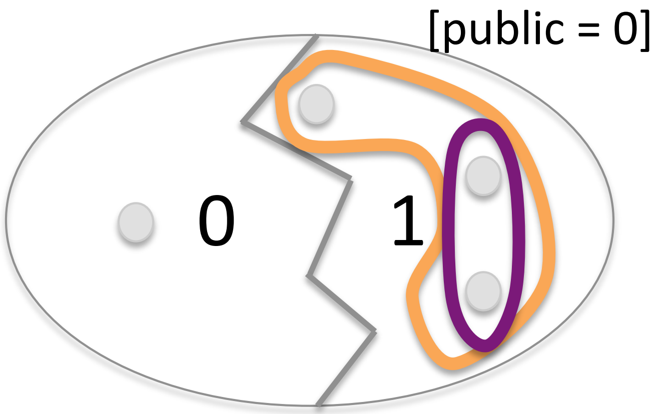
- $(\mathrm{f}, \mathrm{e}) \in \mathbb{F}\times\mathbb{E}$ returns the value of e when f holds in ρ:
  - if $[\![\mathrm{f}]\!]_\rho$ then $[\![\mathrm{e}]\!]_\rho$ else $\top$
- $\phi \in \mathbb{F}$ specifies when the knowledge is $\bot$
  - if $[\![\phi]\!]_\rho$ then $\bot$

Result 1: Correctness guarantee

- Hybrid monitor over-approximates attacker's knowledge

APPROXIMATED KNOWLEDGE

h1= 1



insecure executions

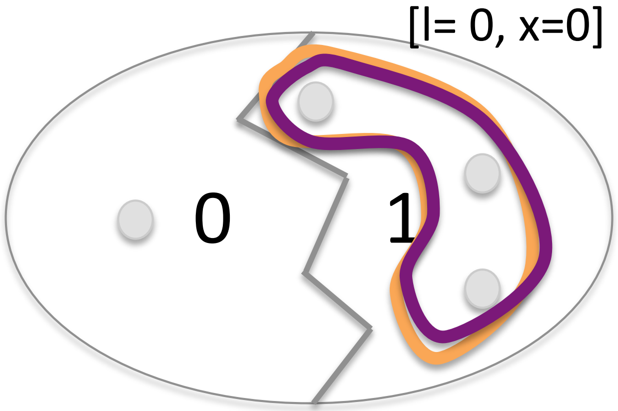
REAL KNOWLEDGE

h1=1 ∨ h2=1

Result 2: Precision

[h1=0, h2=1, l=0, x=0]

```
if h1 = 1 then x = 1
else skip;
if h2 = 1 then l = 1
else l = x;
output l
```



APPROXIMATED KNOWLEDGE

$\boxed{h1 = 1 \lor h2 = 1}$

insecure executions

REAL KNOWLEDGE

$\boxed{h1 = 1 \lor h2 = 1}$

Result 3: Enforcement of noninterference

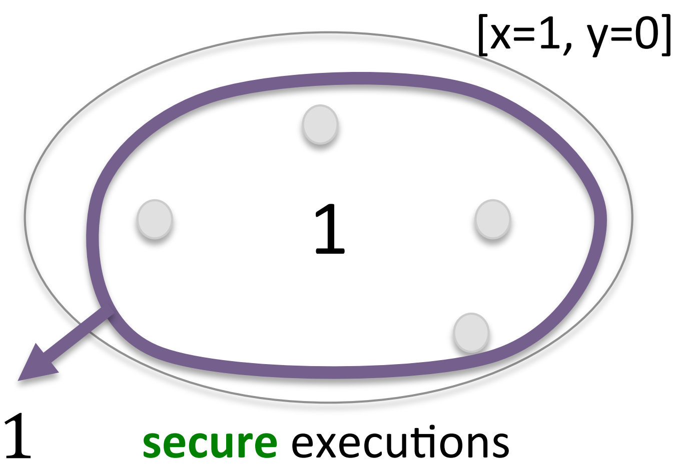
[h=1, x=1, y=0]

```
if h = 1 then l = x + y;
else l = x - y;
output l
```

ACCEPTED

APPROXIMATED KNOWLEDGE

no knowledge

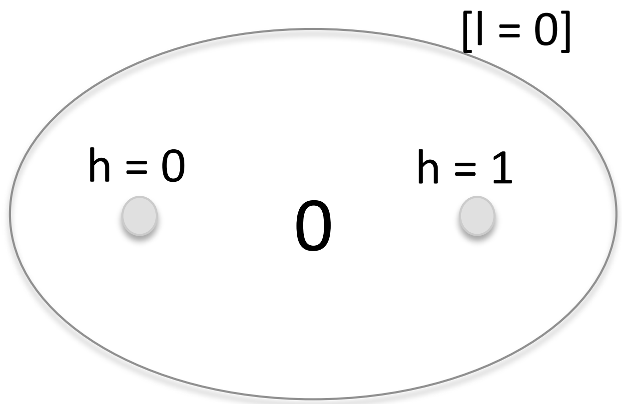


REAL KNOWLEDGE

no knowledge

Result 4: Provably more permissive monitor

```
l = 0;
if h = 1 then skip
else
    x = 5;
    while x > 0 do
        x = x-1; l = x;
output l
```



secure executions

Our monitor combined with inlined dynamic monitor accepts all executions of this secure program

(More details in the paper)

Conclusions

- **Hybrid monitor tracks attacker's knowledge**
  - more precise than [Besson et al. CSF'13]
  - modeled and proved correct
  - enforces noninterference (TINI)
  - has running prototype
- **Combination with another monitor**
  - proved sound (TINI)
  - proved more permissive than previous monitors

Postdoc position

- **Information flow control**
- **Security monitors and type systems**
- **Soundness and permissiveness**

- Starting date: flexible, Nov 2016 – Jun 2017
- Duration: 1 year
- Location: INRIA Sophia Antipolis (Nice, France)

Thank you!

Conclusions

- **Hybrid monitor tracks attacker's knowledge**
  - more precise than [Besson et al.'13]
  - modeled and proved correct
  - enforces noninterference (TINI)
  - has running prototype

- **Combination with another monitor**
  - proved sound (TINI)
  - proved more permissive