Monitoring Information Flow

Dynamic tracking of the information flows at execution time

Gurvan Le Guernic
Thomas Jensen

Université de Rennes 1 - Kansas State University
CNRS

June 30, 2005 / FCS'05

Outline

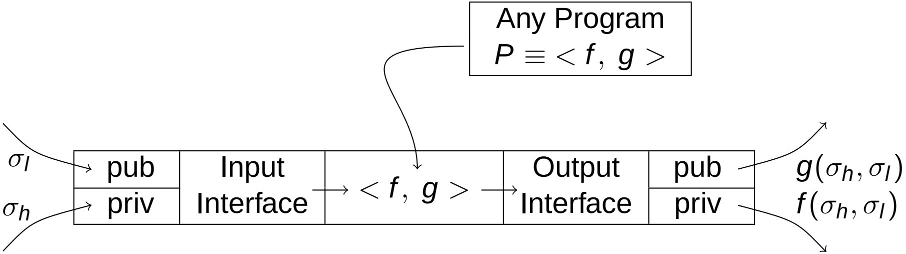
1. Introduction
   - Goal
   - Non-interference
   - Non-interfering execution
2. Detecting Information Flows
   - Semantics
   - Example
3. Correcting Information Flows
   - Main Idea
   - Problem
   - Solution
4. Conclusion

Goal

$P : \Sigma_{H_i} \times \Sigma_{L_i} \rightarrow \Sigma_{H_o} \times \Sigma_{L_o}$

$f : \Sigma_{H_i} \times \Sigma_{L_i} \rightarrow \Sigma_{H_o} = P\downarrow_1$ (private output slice)

$g : \Sigma_{H_i} \times \Sigma_{L_i} \rightarrow \Sigma_{L_o} = P\downarrow_2$ (public output slice)



Any Program

$P \equiv < f, g >$

σ_l

σ_h

pub

priv

Input Interface

$< f, g >$

Output Interface

pub

priv

$g(\sigma_h, \sigma_l)$

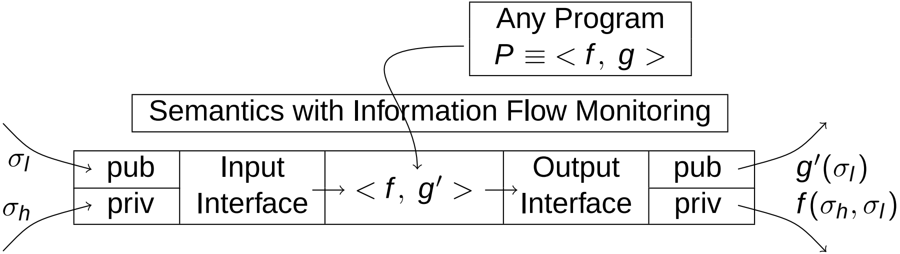
$f(\sigma_h, \sigma_l)$

Goal

$P : \Sigma_{H_i} \times \Sigma_{L_i} \rightarrow \Sigma_{H_o} \times \Sigma_{L_o}$

$f : \Sigma_{H_i} \times \Sigma_{L_i} \rightarrow \Sigma_{H_o} = P\downarrow_1$ (private output slice)

$g : \Sigma_{H_i} \times \Sigma_{L_i} \rightarrow \Sigma_{L_o} = P\downarrow_2$ (public output slice)



Any Program
$P \equiv < f, \; g >$

Semantics with Information Flow Monitoring

σ_l → pub | σ_h → priv

Input Interface → $< f, \; g' >$ → Output Interface

pub → $g'(\sigma_l)$ | priv → $f(\sigma_h, \sigma_l)$

- $\forall o \in \text{dom}(\Sigma_{L_o}), \; \sigma_h \in \Sigma_{H_i}, \; \sigma_l \in \Sigma_{L_i}$:
  $g'(\sigma_l)(o) = g(\sigma_h, \sigma_l)(o) \quad \vee \quad g'(\sigma_l)(o) = \delta$

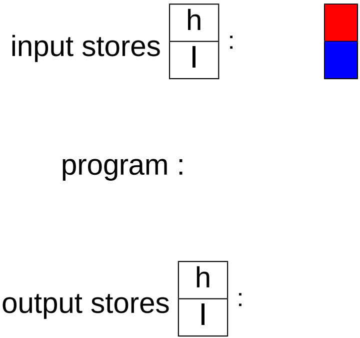
Why not use static analyses ?

- monitoring can be more precise
  - benefit from more contextual information
- allow usage of programs which can not be proved to respect the confidentiality of secret data
  - allow executions proved to respect secrets' confidentiality
  - alter others

Non-Interference

Presentation of the concept of non-interference

- Cohen (77), Goguen and Meseguer (82)
- Property of a program respecting secrets' confidentiality



input stores h l :

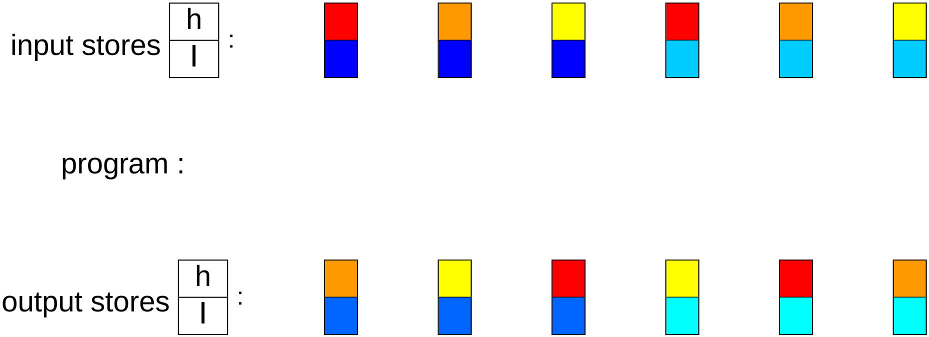
program :

output stores h l :

Non-Interference

Presentation of the concept of non-interference

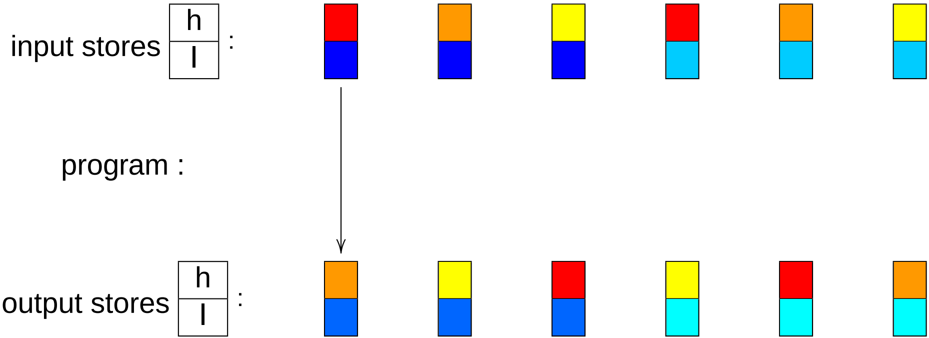
- Cohen (77), Goguen and Meseguer (82)
- Property of a program respecting secrets' confidentiality



Non-Interference

Presentation of the concept of non-interference

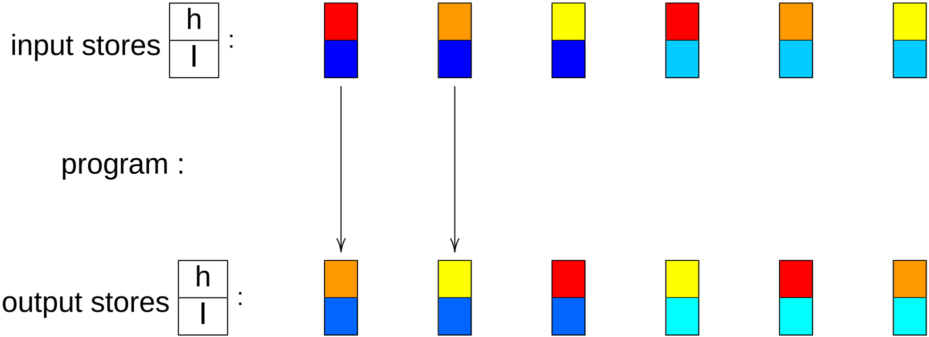
- Cohen (77), Goguen and Meseguer (82)
- Property of a program respecting secrets' confidentiality



Non-Interference

Presentation of the concept of non-interference

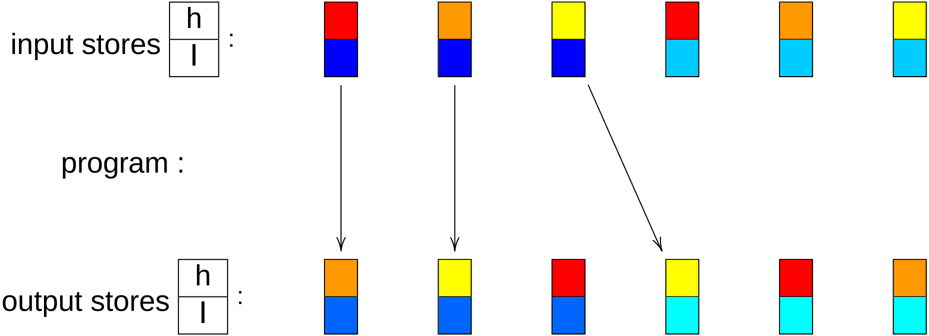
- Cohen (77), Goguen and Meseguer (82)
- Property of a program respecting secrets' confidentiality



Non-Interference

Presentation of the concept of non-interference

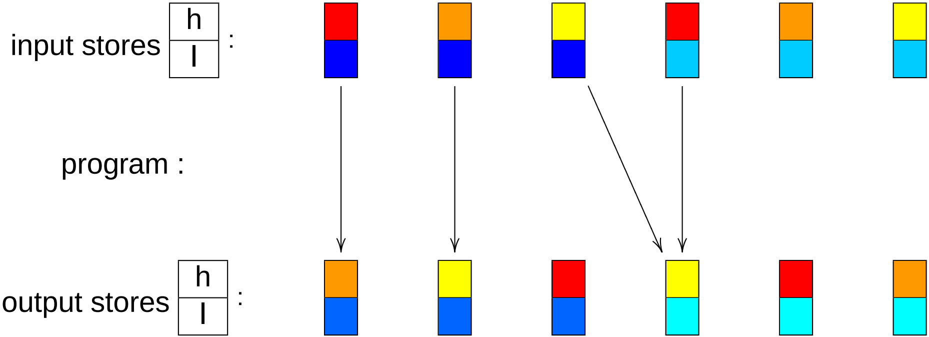
- Cohen (77), Goguen and Meseguer (82)
- Property of a program respecting secrets' confidentiality



Non-Interference

Presentation of the concept of non-interference

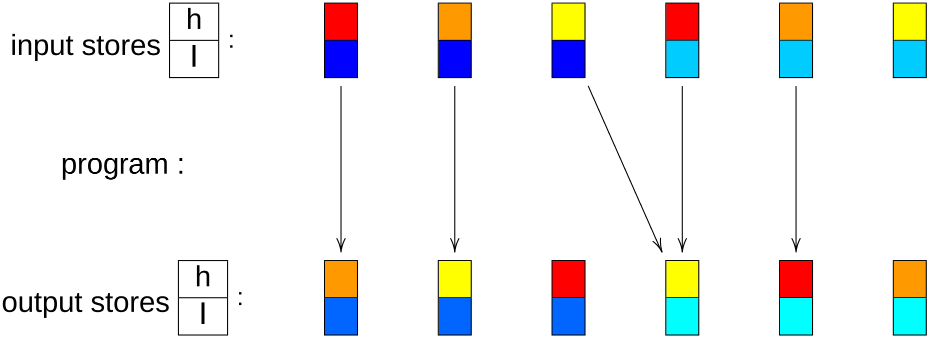
- Cohen (77), Goguen and Meseguer (82)
- Property of a program respecting secrets' confidentiality



Non-Interference

Presentation of the concept of non-interference

- Cohen (77), Goguen and Meseguer (82)
- Property of a program respecting secrets' confidentiality



input stores h l :

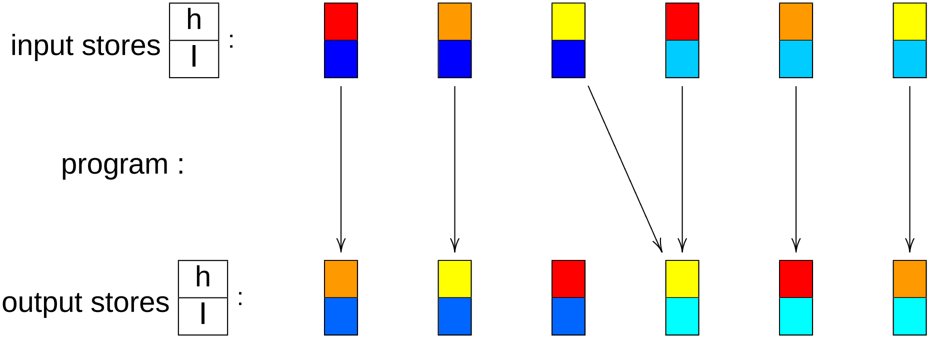
program :

output stores h l :

Non-Interference

Presentation of the concept of non-interference

- Cohen (77), Goguen and Meseguer (82)
- Property of a program respecting secrets’ confidentiality



input stores
h
l
:

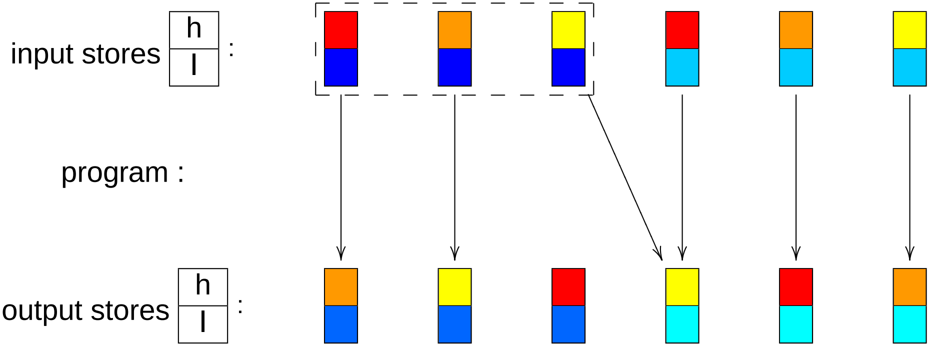
program :

output stores
h
l
:

Non-Interference

Presentation of the concept of non-interference

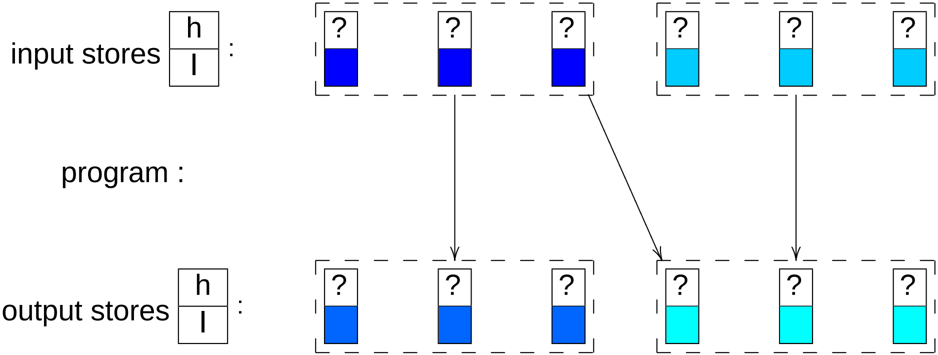
- Cohen (77), Goguen and Meseguer (82)
- Property of a program respecting secrets’ confidentiality



Non-Interference

Presentation of the concept of non-interference

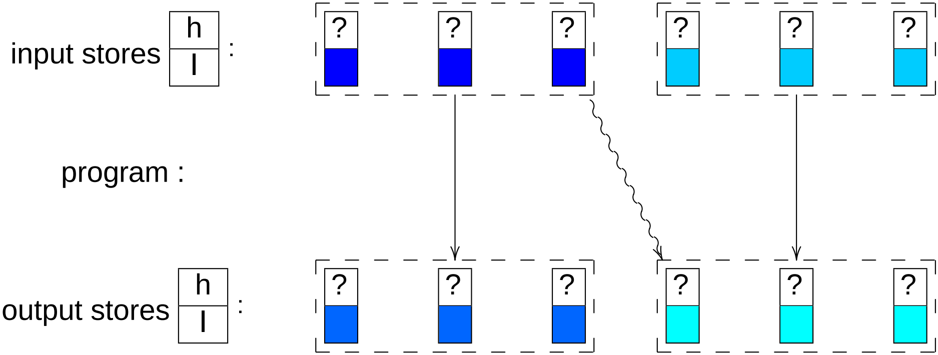
- Cohen (77), Goguen and Meseguer (82)
- Property of a program respecting secrets' confidentiality



Non-Interference

Presentation of the concept of non-interference

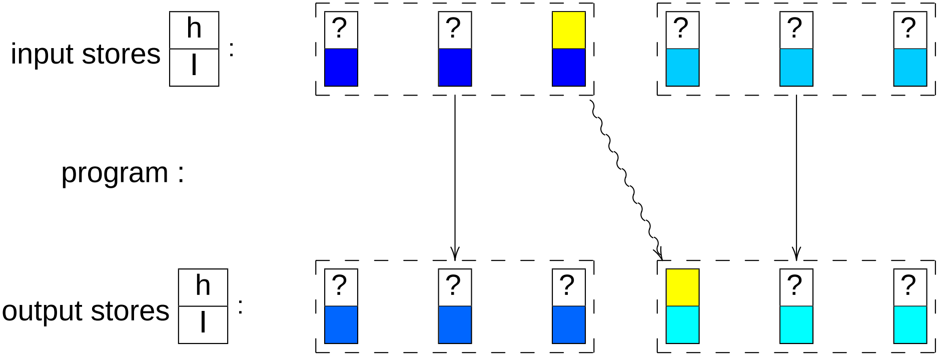
- Cohen (77), Goguen and Meseguer (82)
- Property of a program respecting secrets' confidentiality



Non-Interference

Presentation of the concept of non-interference

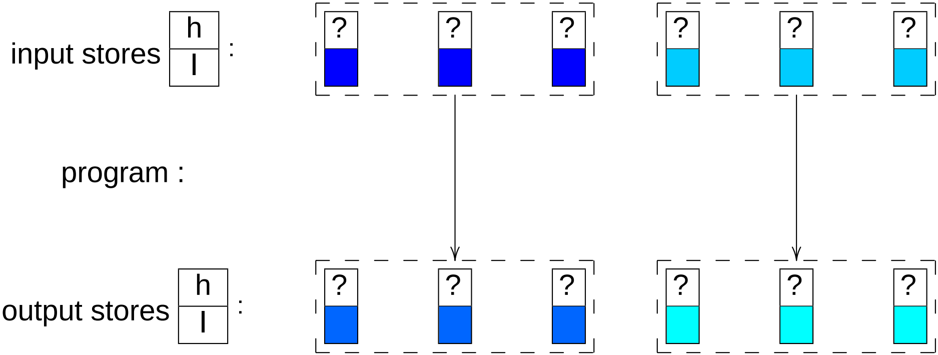
- Cohen (77), Goguen and Meseguer (82)
- Property of a program respecting secrets' confidentiality



Non-Interference

Presentation of the concept of non-interference

- Cohen (77), Goguen and Meseguer (82)
- Property of a program respecting secrets' confidentiality



Non-Interference

Formalization of non-interference

Definition 1 (Non-Interference)

$$\forall s_1, s_2 \in S.\ s_1 =_{L_i} s_2 \Rightarrow [\![C]\!]s_1 =_{L_o} [\![C]\!]s_2$$

- Weaknesses for our purpose :
  - not fitted for monitoring (scope too large)
  - statically difficult to verify precisely

Non-Interference

Formalization of non-interference

Definition 1 (Non-Interference)

$$\forall s_1, s_2 \in S.\ s_1 =_{L_i} s_2 \Rightarrow [\![C]\!]s_1 =_{L_o} [\![C]\!]s_2$$

- Weaknesses for our purpose :
  - not fitted for monitoring (scope too large)
  - statically difficult to verify precisely

Example 2 (Is it possible to deduce the value of h from the one of x ?)

```
x := 0 ; tmp := l ;
if test1( l ) then tmp := h else skip end ;
if test2( l ) then x := tmp else skip end ;
tmp := 0 ; print x ;
```

h : private input
l : public input
x : public output

Non-interfering execution

Main Goal : being able to detect executions respecting the confidentiality of secret data independently from other executions

Non-interfering execution

Main Goal : being able to detect executions respecting the confidentiality of secret data independently from other executions

Definition 3 (Non-interfering execution)

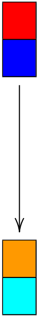
let $s_1 \in S,\ \mathrm{NIExec}(C, s_1) \ \equiv\ \forall s_2.\ s_1 =_{L_i} s_2 \Rightarrow [\![C]\!]s_1 =_{L_o} [\![C]\!]s_2$

Non-interfering execution

Main Goal : being able to detect executions respecting the confidentiality of secret data independently from other executions

Definition 3 (Non-interfering execution)

let $s_1 \in S,\ \mathrm{NIExec}(C, s_1) \equiv \forall s_2.\, s_1 =_{L_i} s_2 \Rightarrow [\![C]\!]s_1 =_{L_o} [\![C]\!]s_2$

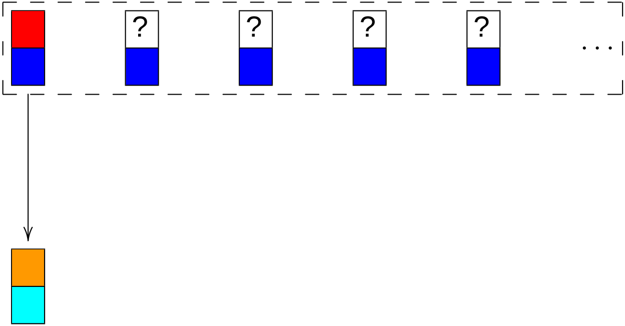


Non-interfering execution

Main Goal : being able to detect executions respecting the confidentiality of secret data independently from other executions

Definition 3 (Non-interfering execution)

let $s_1 \in S,\ \mathrm{NIExec}(C, s_1) \equiv \forall s_2.\ s_1 =_{L_i} s_2 \Rightarrow [\![C]\!]s_1 =_{L_o} [\![C]\!]s_2$

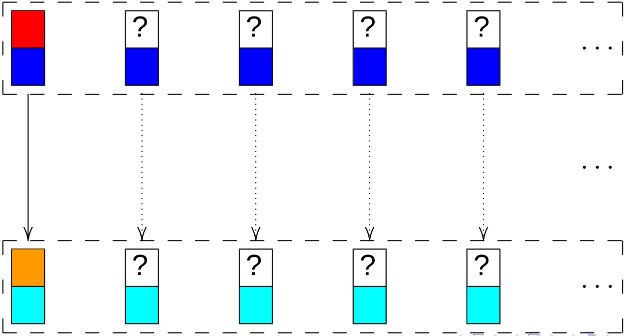


Non-interfering execution

Main Goal : being able to detect executions respecting the confidentiality of secret data independently from other executions

Definition 3 (Non-interfering execution)

let $s_1 \in S,\ \mathrm{NIExec}(C, s_1) \ \equiv \ \forall s_2.\ s_1 =_{L_i} s_2 \Rightarrow [\![C]\!]s_1 =_{L_o} [\![C]\!]s_2$



The approach

- instrumented semantics ($[\![C]\!]$) : $\Sigma_{H_i} \times \Sigma_{L_i} \rightarrow \Sigma^{\mathbb{V}} \times \Sigma^{\mathbb{T}}$
  - value store : $\Sigma^{\mathbb{V}} = \Sigma_{H_o} \times \Sigma_{L_o}$
  - tag store : $\Sigma^{\mathbb{T}}$
- predicate ($\mathrm{Safe}()$) : $\Sigma^{\mathbb{T}} \rightarrow \mathbb{B}$

Assumption 4 (Predicate Safe)

$$\forall s_1 \in S.\ \mathrm{Safe}([\![C]\!]^{\mathbb{T}} s_1) \Rightarrow \mathrm{NIExec}(C, s_1)$$

The approach

- instrumented semantics ($[\![C]\!]$) : $\Sigma_{H_i} \times \Sigma_{L_i} \rightarrow \Sigma^{\mathbb{V}} \times \Sigma^{\mathbb{T}}$
    - value store : $\Sigma^{\mathbb{V}} = \Sigma_{H_o} \times \Sigma_{L_o}$
    - tag store : $\Sigma^{\mathbb{T}}$
- predicate ($\mathrm{Safe}()$) : $\Sigma^{\mathbb{T}} \rightarrow \mathbb{B}$

Assumption 4 (Predicate Safe)

$$\forall s_1 \in S.\ \mathrm{Safe}([\![C]\!]^{\mathbb{T}} s_1) \Rightarrow \mathrm{NIExec}(C, s_1)$$

Proposition 5 (Definition of low-equivalence is symmetric)

$$\forall s_1.\ \mathrm{NIExec}(C, s_1) \Rightarrow (\forall s_2.\ s_2 =_{L_i} s_1 \Rightarrow \mathrm{NIExec}(C, s_2))$$

The approach

- instrumented semantics ($[\![C]\!]$) : $\Sigma_{H_i} \times \Sigma_{L_i} \rightarrow \Sigma^{\mathbb{V}} \times \Sigma^{\mathbb{T}}$
    - value store : $\Sigma^{\mathbb{V}} = \Sigma_{H_o} \times \Sigma_{L_o}$
    - tag store : $\Sigma^{\mathbb{T}}$
- predicate ($\mathrm{Safe}()$) : $\Sigma^{\mathbb{T}} \rightarrow \mathbb{B}$

Assumption 4 (Predicate Safe)

$$\forall s_1 \in S.\ \mathrm{Safe}([\![C]\!]^{\mathbb{T}} s_1) \Rightarrow \mathrm{NIExec}(C, s_1)$$

Proposition 5 (Definition of low-equivalence is symmetric)

$$\forall s_1.\ \mathrm{NIExec}(C, s_1) \Rightarrow (\forall s_2.\ s_2 =_{L_i} s_1 \Rightarrow \mathrm{NIExec}(C, s_2))$$

Benefit : one execution may be sufficient to deduce a property of infinitely many executions

General Description

- general idea :
  - data are tagged ($\bot \sqsubseteq \top$)
    - $\bot \Rightarrow$ data is constant for all σ_L-equivalent executions
    - $\top \Rightarrow$ value may be different

General Description

- general idea :
  - data are tagged ($\bot \sqsubseteq \top$)
    - $\bot \Rightarrow$ data is constant for all σ_L-equivalent executions
    - $\top \Rightarrow$ value may be different
  - instrumented semantics updates tags

General Description

- general idea :
  - data are tagged ($\bot \sqsubseteq \top$)
    - $\bot \Rightarrow$ data is constant for all σ_L-equivalent executions
    - $\top \Rightarrow$ value may be different
  - instrumented semantics updates tags
  - $\mathrm{Safe}(\llbracket C \rrbracket s_1)$ **iff** public outputs are tagged with $\bot$

General Description

- general idea :
  - data are tagged ($\bot \sqsubseteq \top$)
    - $\bot \Rightarrow$ data is constant for all σ_L-equivalent executions
    - $\top \Rightarrow$ value may be different
  - instrumented semantics updates tags
  - $\mathrm{Safe}(\llbracket C \rrbracket s_1)$ **iff** public outputs are tagged with $\bot$

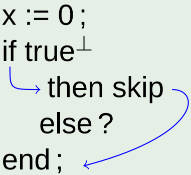
Example 6

```
x := 0 ;
if true$^{\bot}$
      then skip
     else ?
end ;
```

General Description

- general idea :
  - data are tagged ($\bot \sqsubseteq \top$)
    - $\bot \Rightarrow$ data is constant for all σ_L-equivalent executions
    - $\top \Rightarrow$ value may be different
  - instrumented semantics updates tags
  - $\mathrm{Safe}(\llbracket C \rrbracket s_1)$ **iff** public outputs are tagged with $\bot$

Example 6



x := 0 ;

if true$^{\bot}$

then skip

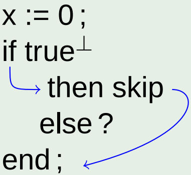
else ?

end ;

General Description

- general idea :
  - data are tagged ($\perp \sqsubseteq \top$)
    - $\perp \Rightarrow$ data is constant for all σ_L-equivalent executions
    - $\top \Rightarrow$ value may be different
  - instrumented semantics updates tags
  - $\mathrm{Safe}(\llbracket C \rrbracket s_1)$ **iff** public outputs are tagged with $\perp$

Example 6



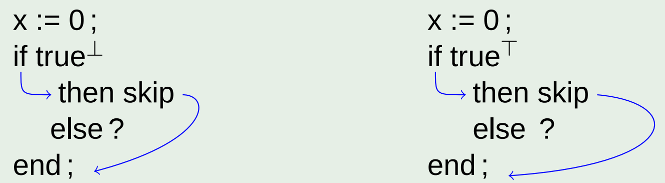
x := 0 ;
if true$^{\perp}$
    then skip
  else ?
end ;

x := 0 ;
if true$^{\top}$
    then skip
    else ?
end ;

General Description

- general idea :
  - data are tagged ($\bot \sqsubseteq \top$)
    - $\bot \Rightarrow$ data is constant for all σ_L-equivalent executions
    - $\top \Rightarrow$ value may be different
  - instrumented semantics updates tags
  - $\mathrm{Safe}(\llbracket C \rrbracket s_1)$ **iff** public outputs are tagged with $\bot$

Example 6



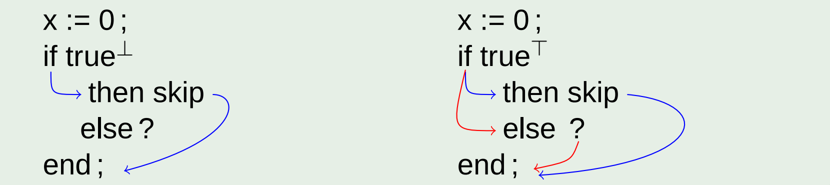
```
x := 0 ;
if true^⊥
  then skip
  else ?
end ;
```

```
x := 0 ;
if true^⊤
  then skip
  else ?
end ;
```

General Description

- general idea :
  - data are tagged ($\perp \sqsubseteq \top$)
    - $\perp \Rightarrow$ data is constant for all σ_L-equivalent executions
    - $\top \Rightarrow$ value may be different
  - instrumented semantics updates tags
  - $\mathrm{Safe}([\![C]\!]s_1)$ **iff** public outputs are tagged with $\perp$

Example 6



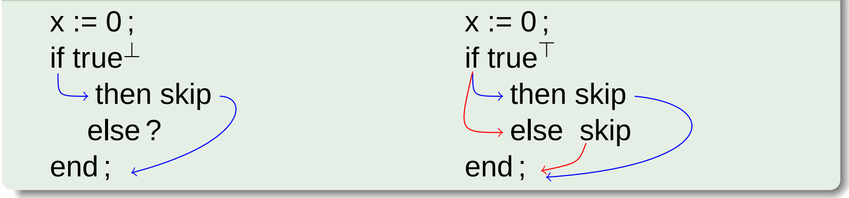
```
x := 0 ;
if true$^\perp$
    then skip
    else ?
end ;
```

```
x := 0 ;
if true$^\top$
    then skip
    else ?
end ;
```

General Description

- general idea :
  - data are tagged ($\perp \sqsubseteq \top$)
    - $\perp \Rightarrow$ data is constant for all σ_L-equivalent executions
    - $\top \Rightarrow$ value may be different
  - instrumented semantics updates tags
  - $\mathrm{Safe}(\llbracket C \rrbracket s_1)$ **iff** public outputs are tagged with $\perp$

Example 6



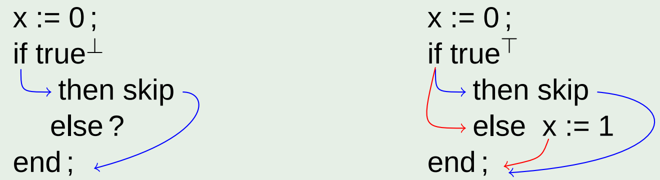
x := 0 ;
if $\mathrm{true}^{\perp}$
then skip
else ?
end ;

x := 0 ;
if $\mathrm{true}^{\top}$
then skip
else skip
end ;

General Description

- general idea :
  - data are tagged ($\bot \sqsubseteq \top$)
    - $\bot \Rightarrow$ data is constant for all σ_L-equivalent executions
    - $\top \Rightarrow$ value may be different
  - instrumented semantics updates tags
  - $\mathrm{Safe}([\![C]\!]s_1)$ **iff** public outputs are tagged with $\bot$

Example 6



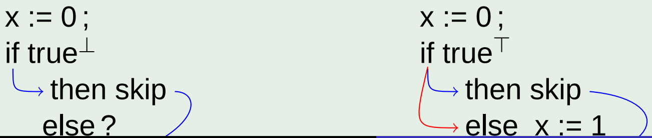
x := 0 ;
if true$^{\bot}$
then skip
else ?
end ;

x := 0 ;
if true$^{\top}$
then skip
else x := 1
end ;

General Description

- general idea :
  - data are tagged ($\perp \sqsubseteq \top$)
    - $\perp \Rightarrow$ data is constant for all σ_L-equivalent executions
    - $\top \Rightarrow$ value may be different
  - instrumented semantics updates tags
  - $\mathrm{Safe}(\llbracket C \rrbracket s_1)$ **iff** public outputs are tagged with $\perp$
- when branching on a condition which is :
  - $\perp$: execute the designated branch
  - $\top$: merge the result of the execution of the designated branch with an analysis that approximates all the other σ_L-equivalent executions that take the other branch

Example 6



x := 0 ;
if true$^{\perp}$
then skip
else ?

x := 0 ;
if true$^{\top}$
then skip
else x := 1

Properties of the semantics

Definition 7 (Predicate Safe)

For all tag store $\rho \in \Sigma^{\mathbb{T}}(= Var \rightarrow \{\bot, \top\})$, $\mathrm{Safe}(\rho)$ iff :

$$\forall o \in \mathrm{dom}(\Sigma_{L_o}),\ \rho(o) = \bot$$

Theorem 8

For any command C and $\sigma_1, \sigma_2 \in \Sigma_{H_i} \times \Sigma_{L_i}$, such that :

1. $[\![C]\!]^{\mathbb{V}}_{\sigma_2} \neq \bot$
2. $\mathrm{Safe}([\![C]\!]^{\mathbb{T}}_{\sigma_1})$

$$\textit{if } \sigma_1 =_{L_i} \sigma_2 \textit{ then } [\![C]\!]^{\mathbb{V}}_{\sigma_1} =_{L_o} [\![C]\!]^{\mathbb{V}}_{\sigma_2}$$

Example

Example 9

```
x := 0 ;
if l then
    if h then x := 1 else skip end
else skip end
```

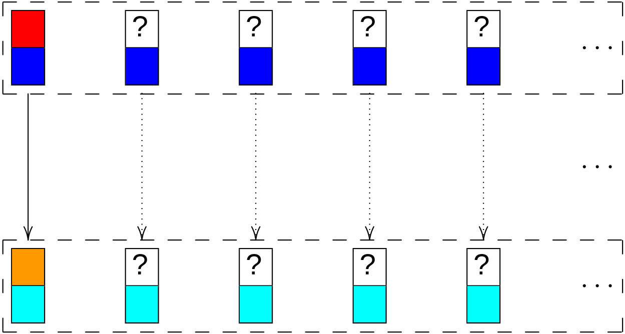
Final value of x

σ_l(l) σ_h(h)	True	False
True	1	0
False	0	0

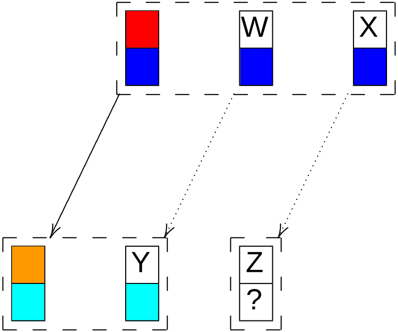
Final tag of x

σ_l(l) σ_h(h)	True	False
True	$\top$	$\bot$
False	$\top$	$\bot$

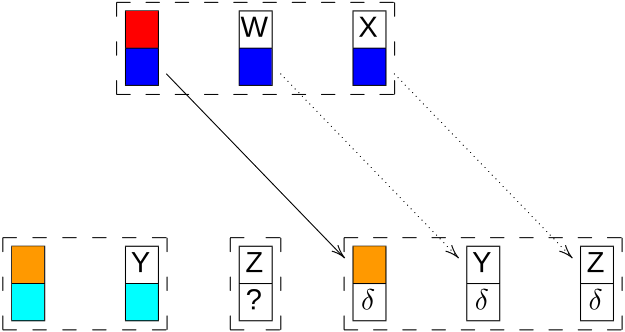
Correcting Information Flows : Main Idea



Correcting Information Flows : Main Idea



Correcting Information Flows : Main Idea



- set a default value (δ) for *unsafe* public outputs
- keep the value of private outputs and *safe* public outputs

Problem : correction may cause information leakage

Fact 10

$$\forall s_1, s_2 \in S.\ s_1 =_{L_i} s_2 \not\Rightarrow (\mathrm{Safe}([\![C]\!]^{\mathbb{T}} s_1) \Leftrightarrow \mathrm{Safe}([\![C]\!]^{\mathbb{T}} s_2))$$

Problem : correction may cause information leakage

Fact 10

$$\forall s_1, s_2 \in S.\ s_1 =_{L_i} s_2 \not\Rightarrow (\mathrm{Safe}(\llbracket C \rrbracket^{\mathbb{T}} s_1) \Leftrightarrow \mathrm{Safe}(\llbracket C \rrbracket^{\mathbb{T}} s_2))$$

Example 11

```
x := 0 ;
if h then
    if l then x := 1 else skip end
else skip end
```

Final value of x

σ_l(l) σ_h(h)	True	False
True	1	0
False	0	0

Final tag of x

σ_l(l) σ_h(h)	True	False
True	$\top$	$\bot$
False	$\top$	$\top$

Problem : correction may cause information leakage

Fact 10

$$\forall s_1, s_2 \in S.\ s_1 =_{L_i} s_2 \not\Rightarrow (\mathrm{Safe}(\llbracket C \rrbracket^{\mathbb{T}} s_1) \Leftrightarrow \mathrm{Safe}(\llbracket C \rrbracket^{\mathbb{T}} s_2))$$

Example 11

```
x := 0 ;
if h then
   if l then x := 1 else skip end
else skip end
```

Final value of x

σ_h(h) \ σ_l(l)	True	False
True	δ	0
False	δ	δ

Final tag of x

σ_h(h) \ σ_l(l)	True	False
True	$\top$	$\bot$
False	$\top$	$\top$

Work in progress

Theorem 12 (proof in progress)

For any command C and $\sigma_1, \sigma_2 \in \Sigma_{H_i} \times \Sigma_{L_i}$, *such that :*

1. $[\![C]\!]^{\mathbb{V}}_{\sigma_1} \neq \bot$ *and* $[\![C]\!]^{\mathbb{V}}_{\sigma_2} \neq \bot$

$$\textit{if } \sigma_1 =_{L_i} \sigma_2 \textit{ then } [\![C]\!]^{\mathbb{T}}_{\sigma_1} =_{L_o} [\![C]\!]^{\mathbb{T}}_{\sigma_2}$$

Work in progress

Theorem 12 (proof in progress)

For any command C and $\sigma_1, \sigma_2 \in \Sigma_{H_i} \times \Sigma_{L_i}$*, such that :*

1. $[\![C]\!]^{\mathbb{V}}_{\sigma_1} \neq \bot$ *and* $[\![C]\!]^{\mathbb{V}}_{\sigma_2} \neq \bot$

$$\textit{if } \sigma_1 =_{L_i} \sigma_2 \textit{ then } [\![C]\!]^{\mathbb{T}}_{\sigma_1} =_{L_o} [\![C]\!]^{\mathbb{T}}_{\sigma_2}$$

Hypothesis 13 (proof in progress)

“$[\![\sigma ; \rho \vdash C]\!]^{\sharp\mathcal{G}}$ *and* $[\![C]\!]^{\mathbb{T}}_{\sigma}$ *compute the same tags*”

Related work

- M. Abadi, B. Lampson, and J.-J. Lévy.
  Analysis and caching of dependencies.
  In *Proc. ACM International Conf. on Functional Programming*, pages 83–91, 1996.

- A. C. Myers.
  JFlow : Practical mostly-static information flow control.
  In *Proc. ACM Symp. Principles of Programming Languages*, pages 228–241, 1999.

- D. Volpano, G. Smith, and C. Irvine.
  A sound type system for secure flow analysis.
  J. Computer Security, 4(3) :167–187, 1996.

- A. Sabelfeld and A. C. Myers.
  Language-based information-flow security.
  IEEE J. Selected Areas in Communications, 21(1) :5–19, Jan. 2003.

Final word

- A non-interference definition with a reduced scope :
  - non-interfering execution
- A “smart” semantics
- A predicate for detecting non-interfering executions
- A method for dynamic correction of potential leakages

Final word

- A non-interference definition with a reduced scope :
  - non-interfering execution
- A “smart” semantics
- A predicate for detecting non-interfering executions
- A method for dynamic correction of potential leakages

⇒ Possible to detect the “safe” behavior of a set of executions from only one of those executions ; and correct it if that is not the case.

Monitoring Information Flow

Dynamic tracking of the information flows at execution time

Gurvan Le Guernic
Thomas Jensen

Université de Rennes 1 - Kansas State University
CNRS

June 30, 2005 / FCS'05