

# Security by logic : characterizing Non-Interference in temporal logic

Henri-Charles Blondeel

<sup>1</sup>KTH (Royal Institute of Technology, Stockholm)  
School of Computer Science and Communication  
Master's project

<sup>2</sup>INRIA Sophia-Antipolis, Team EVEREST

December 12, 2007

Supervisor at INRIA : Marieke Huisman

# Motivation

## Intuitive definition of Non-Interference

L and H security levels.

*“The final values of low variables should be independent of the initial values of high variables.”*

## Non-Interference for concurrent programs

Still no consensus on a definition in the setting of concurrency.

## Verifying Non-Interference

Verification method which is automatic and precise.

# Outline

## Comparing different proposals of Non-Interference

- Uniform program model

- Re-expressing proposals over our model

- Comparison of the proposals

## Enforcing Non-Interference by temporal logic

- Security type system vs model checking

- Modal  $\mu$ -calculus

- Characterizing Non-Interference properties in modal  $\mu$ -calculus

- Model checking with the Concurrency Workbench (CWB)

# Outline

## Comparing different proposals of Non-Interference

### Uniform program model

Re-expressing proposals over our model

Comparison of the proposals

## Enforcing Non-Interference by temporal logic

Security type system vs model checking

Modal  $\mu$ -calculus

Characterizing Non-Interference properties in modal  $\mu$ -calculus

Model checking with the Concurrency Workbench (CWB)

# Uniform program model

## Important features

- ▶ Defined by a small step operational semantics
  - ▶ used in many definitions
  - ▶ allows the observation of the intermediate states of an execution
- ▶ Also supports definitions given in terms of process algebra

# Uniform program model

## Language

Simple parallel programming language.

$$S ::= x := E \mid S; S \mid \text{if } (b) \text{ then } S \text{ else } S \mid \\ \text{while } (b) \{ S \} \mid S \parallel_p S \mid \text{wait } (b) \mid \epsilon$$

## Memory

Single shared global memory.

L-equivalence : equality on low store, denoted  $\mu \approx_L \mu'$

# Uniform program model

## Actions

$$\text{Actions} = \{ \text{write}_{v,x} \mid v \in \text{Var}, x \in \text{dom}(v) \} \\ \cup \{ \text{read}_{v,x} \mid v \in \text{Var}, x \in \text{dom}(v) \}$$

## Example (histories)

Execution starting from  $x := y + z$  in store  $\mu$

$$\text{hist} = \{ \text{read}_{y,\mu(y)}, \text{read}_{z,\mu(z)} \}, \{ \text{write}_{x,\mu(y)+\mu(z)} \}$$

## Extending configurations with histories of actions

$$c = \langle S, \mu, \text{hist} \rangle$$

# Uniform program model

## Particularities of our operational semantics

- ▶ It specifies how histories are updated.

### Example (assign rule)

---

$$\langle x := E, \mu, \text{hist} \rangle \rightarrow \langle \epsilon, \mu(x \rightarrow E(\mu)), \text{hist} \hat{=} \text{readV}(E, \mu) \hat{=} \{\text{write}_{x, E(\mu)}\} \rangle$$

- ▶ The term and deadl rules ensure that we can always take a new transition.

### Example (term rule)

---

$$\langle \epsilon, \mu, \text{hist} \rangle \rightarrow \langle \epsilon, \mu, \text{hist} \rangle$$

# Uniform program model

## Traces of executions

$T = c_0 c_1 c_2 \dots$  such that  $\forall i. c_i \rightarrow c_{i+1}$   
denoted  $c_0 \Downarrow T$

We use  $T_i$  to denote the  $(i + 1)^{th}$  configuration of  $T$ .

## Advantages of this notation

- ▶ simple description of executions
- ▶ notions as termination, reachability, or equivalence of executions can be defined in terms of traces

## Reachability

$$\text{reach}(S, \mu, \text{hist}) = \{T_i \in \text{Config} \mid \langle S, \mu \rangle \Downarrow T \wedge T(\text{hist})_i = \text{hist}\}$$

# Outline

## Comparing different proposals of Non-Interference

Uniform program model

Re-expressing proposals over our model

Comparison of the proposals

## Enforcing Non-Interference by temporal logic

Security type system vs model checking

Modal  $\mu$ -calculus

Characterizing Non-Interference properties in modal  $\mu$ -calculus

Model checking with the Concurrency Workbench (CWB)

# Observational determinism

## Authors

Introduced by Zdancewic and Myers (2003) - refined by Huisman et al. (2006)

## About observational determinism

- ▶ It characterizes, for all low variables, the equivalence of all possible *location traces* up to stuttering.
- ▶ For race-free programs only
- ▶ Stuttering equivalence can be defined in terms of traces, denoted  $T(I) \sim_s T'(I)$  (recursive definition).

# Observational determinism

## Definition (Observational determinism)

A program  $S$  is observationally deterministic if

$$\forall \mu, \mu' \in \text{Store}. \forall T, T' \in \text{Trace}. \\ \mu \approx_L \mu' \wedge \langle S, \mu \rangle \Downarrow T \wedge \langle S, \mu' \rangle \Downarrow T' \Rightarrow T \approx_L T'$$

▶ (charac.)

## Remark

Our program model subsumes Huisman et al.'s model, thus we have identical definitions.

# Eager and lazy invariance

## Author

Roscoe (1995)

## Definition in CSP

- ▶ In the traces model of CSP, a process  $P$  is identified by the set of finite sequences of actions it can communicate, denoted  $Seq(P)$ .
- ▶ Non-Interference is defined as determinism of the low behaviours.
- ▶ Eager and lazy invariance have different ways to prevent H observations from the attacker.

# Eager and lazy invariance

## Definition (Eager invariance (in CSP))

A process  $P$  is eagerly trace-invariant if

$$s, s' \in \text{Seq}(P) \wedge s \upharpoonright L = s' \upharpoonright L \Rightarrow \text{Seq}((P/s) \setminus H) = \text{Seq}((P/s') \setminus H)$$

## Remarks

- ▶ A variant, lazy invariance, makes  $H$  actions ambiguous in the conclusion instead of hiding them. This is expressed by interleaving with arbitrary  $H$  actions.
- ▶ The reformulation in terms of traces uses the notion of reachability.

# Eager and lazy invariance

We instantiate action sequences as histories defined in our uniform model.

Definition (Eager invariance (in our model))

$$\begin{aligned} \forall \mathit{hist}, \mathit{hist}' \in \mathit{Hist}. \mathit{hist} =_{|L\text{-Actions}} \mathit{hist}' \Rightarrow \\ \forall c \in \mathit{reach}(S, \mathit{Store}, \mathit{hist}). \forall T \in \mathit{Trace}. c \Downarrow T \Rightarrow \\ \exists c' \in \mathit{reach}(S, \mathit{Store}, \mathit{hist}'). \exists T' \in \mathit{Trace}. \\ c' \Downarrow T' \wedge (\forall m \in \mathbb{N}. \exists n \in \mathbb{N}. T(\mathit{hist})_m =_{L\text{-Actions}} T'(\mathit{hist}')_n) \end{aligned}$$

▶ (charac.) We obtain an existential property.

# Non-Interference in terms of bisimulation

## Authors

Sabelfeld and Sands (2000)

## Partial probabilistic bisimulation on configurations

For all equivalence class  $A$ , two related configurations must have the same probability to reach  $A$ .

## Partial probabilistic low bisimulation on statements

$\sim \times \approx_L$  is a partial probabilistic bisimulation on configurations.

# NI in terms of bisimulation

## Definition (Non-Interference based on low bisimulation)

A program  $S$  is secure *w.r.t.*  $L$  if there exists a partial probabilistic low-bisimulation on statements  $\sim_L$  such that  $S \sim_L S$ .

## Theorem

*A per  $R$  is a partial probabilistic low-bisimulation on Stmt iff*

$$\begin{aligned} & \forall S_1, S_2 \in \text{Stmt}. \forall \mu_1, \mu_2 \in \text{Store}. \\ & (S_1 R S_2 \wedge \mu_1 \approx_L \mu_2 \wedge \langle S_1, \mu_1 \rangle \rightarrow \langle S'_1, \mu'_1 \rangle) \Rightarrow \\ & \quad \exists S'_2 \in \text{Stmt}, \mu'_2 \in \text{Store}. \\ & \quad \langle S_2, \mu_2 \rangle \rightarrow \langle S'_2, \mu'_2 \rangle \wedge \\ & \quad S'_1 R S'_2 \wedge \\ & \quad \mu'_1 \approx_L \mu'_2 \wedge \\ & \quad \sum_{S \in [S'_1]_R, \mu \approx_L \mu'_1} p_\sigma(\langle S_1, \mu_1 \rangle \rightarrow \langle S, \mu \rangle) = \\ & \quad \sum_{S \in [S'_2]_R, \mu \approx_L \mu'_2} p_\sigma(\langle S_2, \mu_2 \rangle \rightarrow \langle S, \mu \rangle) \end{aligned}$$

## Other proposals

- ▶ Probabilistic Non-Interference, Volpano and Smith (1998)
  - ▶ probabilistic definition
  - ▶ for finite state spaces only
- ▶ Abstract interpretation, Giacobazzi and Mastroeni (2004)
  - ▶ takes into account what the attacker can observe
  - ▶ weaker than Non-Interference
  - ▶ for sequential programs only
- ▶ Non-Interference in the knowledge setting, Askarov and Sabelfeld (2007)
  - ▶ defines and uses the knowledge of an attacker about the initial store
  - ▶ can be extended to declassification

# Outline

## Comparing different proposals of Non-Interference

Uniform program model

Re-expressing proposals over our model

Comparison of the proposals

## Enforcing Non-Interference by temporal logic

Security type system vs model checking

Modal  $\mu$ -calculus

Characterizing Non-Interference properties in modal  $\mu$ -calculus

Model checking with the Concurrency Workbench (CWB)

# Comparison of the different proposals

## Possibilistic vs probabilistic definitions

Possibilistic definitions cannot prevent an attacker from learning information by doing a probabilistic analysis.

## Main characteristics

- ▶ Compositionality
- ▶ Termination sensitive or insensitive
- ▶ Granularity

# Comparison for a set of examples

## Example (Observational determinism and eager invariance)

$$I := true \parallel_p I := false$$

is eagerly invariant but not observationally deterministic.

$$I := true; \text{if } (h) \text{ then } (\text{if } (I) \text{ then } \epsilon \text{ else } \epsilon) \text{ else } \epsilon$$

is observationally deterministic but not eagerly invariant.

## Conclusion

All properties are uncomparable

# Outline

## Comparing different proposals of Non-Interference

- Uniform program model

- Re-expressing proposals over our model

- Comparison of the proposals

## Enforcing Non-Interference by temporal logic

- Security type system vs model checking

- Modal  $\mu$ -calculus

- Characterizing Non-Interference properties in modal  $\mu$ -calculus

- Model checking with the Concurrency Workbench (CWB)

# Security type system vs model checking

## Security type systems

- ▶ A program is accepted if it is typable.
- ▶ This method is not precise (syntactic equality).
- ▶ Most studied approach

## Temporal logics and model checking

- ▶ Basics
- ▶ Already investigated by Huisman et al. (CTL\* characterization of observational determinism)
- ▶ Expressive power of  $\mu$ -calculus

# Outline

## Comparing different proposals of Non-Interference

Uniform program model

Re-expressing proposals over our model

Comparison of the proposals

## Enforcing Non-Interference by temporal logic

Security type system vs model checking

**Modal  $\mu$ -calculus**

Characterizing Non-Interference properties in modal  $\mu$ -calculus

Model checking with the Concurrency Workbench (CWB)

# Modal $\mu$ -calculus

## Syntax

Let  $Act$  be a set of actions labels, ranged over by  $\alpha$ .

$$\Phi ::= \text{tt} \mid \text{ff} \mid X \mid \neg\Phi \mid \Phi \wedge \Phi \mid \langle \alpha \rangle \Phi \mid [\alpha] \Phi \mid \mu X. \Phi \mid \nu X. \Phi$$

$$\begin{array}{ll} \mu X^0. \Phi & \stackrel{\text{def}}{=} \text{ff} & \nu X^0. \Phi & \stackrel{\text{def}}{=} \text{tt} \\ \mu X^{k+1}. \Phi & \stackrel{\text{def}}{=} \Phi[\mu X^k. \Phi / X] & \nu X^{k+1}. \Phi & \stackrel{\text{def}}{=} \Phi[\nu X^k. \Phi / X] \end{array}$$

## Semantics

Let  $T = (\mathcal{S}, Act, \rightarrow)$  be a finite state labelled transition system.

$$\begin{array}{l} \dots \\ s \models^T \langle \alpha \rangle \Phi \quad \stackrel{\text{def}}{\Leftrightarrow} \quad \exists s' \in \mathcal{S}. (s \xrightarrow{\alpha} s' \wedge s' \models^T \Phi) \\ s \models^T [\alpha] \Phi \quad \stackrel{\text{def}}{\Leftrightarrow} \quad \forall s' \in \mathcal{S}. (s \xrightarrow{\alpha} s' \Rightarrow s' \models^T \Phi) \\ s \models^T \mu X. \Phi \quad \stackrel{\text{def}}{\Leftrightarrow} \quad \exists k \in \mathbb{N}. s \models^T \mu X^k. \Phi \\ s \models^T \nu X. \Phi \quad \stackrel{\text{def}}{\Leftrightarrow} \quad \forall k \in \mathbb{N}. s \models^T \nu X^k. \Phi \end{array}$$

# Modal $\mu$ -calculus : informal semantics

## Alternation-free formulae

$\nu$  means looping (or safety) and  $\mu$  finite looping (or liveness)

## Example (always $\phi$ )

$$\nu X. \phi \wedge [\alpha] X = \phi \wedge [\alpha] (\phi \wedge [\alpha] (\phi \wedge [\alpha] (\dots)))$$

## Example (possible $\psi$ )

$$\mu Y. \psi \vee \langle \alpha \rangle Y = \psi \vee \langle \alpha \rangle (\psi \vee \langle \alpha \rangle (\dots (\psi \vee \langle \alpha \rangle \text{ff})))$$

## Example (Expressive power of $\mu$ -calculus)

$$\nu X. (\mu Y. \psi \vee \langle \alpha \rangle Y) \wedge [\alpha] X$$

# Outline

## Comparing different proposals of Non-Interference

Uniform program model

Re-expressing proposals over our model

Comparison of the proposals

## Enforcing Non-Interference by temporal logic

Security type system vs model checking

Modal  $\mu$ -calculus

**Characterizing Non-Interference properties in modal  $\mu$ -calculus**

Model checking with the Concurrency Workbench (CWB)

# Characterizing observational determinism

Finding an appropriate labelled transition system

$$T = (\mathcal{S}, Act, \rightarrow)$$

Self-composition

▶ (definition)

Labelled transitions for a single program execution

$$\frac{\mu(x) \neq E(\mu) \quad v = E(\mu)}{\langle x := E, \mu \rangle \xrightarrow{c_{x,v}} \langle \epsilon, \mu(x \rightarrow v) \rangle}$$
$$\frac{\mu(x) = E(\mu)}{\langle x := E, \mu \rangle \xrightarrow{\tau} \langle \epsilon, \mu(x \rightarrow E(\mu)) \rangle}$$

# Characterizing observational determinism : LTS

Set of states  $\mathcal{S}$

$$\mathcal{S} = \text{Config} \times \text{Config}$$

Set of actions  $\text{Act}$

$$\text{Act} = \{(a)_j \mid a \in \text{Action}_{OD} \wedge j \in \{1, 2\}\}$$

Labelled transition relation  $\rightarrow$

$$\frac{\langle S_1, \mu_1 \rangle \xrightarrow{a} \langle S'_1, \mu'_1 \rangle}{\langle \langle S_1, \mu_1 \rangle, \langle S_2, \mu_2 \rangle \rangle \xrightarrow{(a)_1} \langle \langle S'_1, \mu'_1 \rangle, \langle S_2, \mu_2 \rangle \rangle}$$
$$\frac{\langle S_2, \mu_2 \rangle \xrightarrow{a} \langle S'_2, \mu'_2 \rangle}{\langle \langle S_1, \mu_1 \rangle, \langle S_2, \mu_2 \rangle \rangle \xrightarrow{(a)_2} \langle \langle S_1, \mu_1 \rangle, \langle S'_2, \mu'_2 \rangle \rangle}$$

# Characterizing observational determinism : formula

## $\mu$ -calculus characterization

$$\Phi_{OD} = (\bigwedge_{l \in \text{Var}_L} \text{eq}_l) \Rightarrow \bigwedge_{l \in \text{Var}_L} \Psi_l$$

$$\Psi_l = \nu R. \text{ always}^{(\bar{c}_l)_1} ([c_l]_1) ( \text{ finally}^{(-)_2^l} (\text{eq}_l) \wedge \text{ always}^{(\bar{c}_l)_2} ([c_l]_2) (\text{eq}_l \wedge R) ) )$$

## abbreviations

$$\text{always}^A(\phi) = \nu X. \phi \wedge \left( \bigwedge_{a \in A} [a] X \right)$$

$$\text{finally}^A(\phi) = \mu X. \phi \vee \left( \bigwedge_{a \in A} [a] X \right)$$

# Characterizing eager invariance

## Labelled transitions for a single program execution

For simplicity, we assume that histories are sequences of actions.

$$\frac{\langle S, \mu, \text{hist} \rangle \rightarrow \langle S', \mu', \text{hist} \rangle}{\langle S, \mu, \text{hist} \rangle \xrightarrow{\tau} \langle S', \mu', \text{hist} \rangle}$$
$$\frac{\langle S, \mu, \text{hist} \rangle \rightarrow \langle S', \mu', \text{hist} \wedge \{a\} \rangle}{\langle S, \mu, \text{hist} \rangle \xrightarrow{a} \langle S', \mu', \text{hist} \wedge \{a\} \rangle}$$

## Triple program model

Our labelled transition system describes three executions.

▶ (definition)

## Initialization issue

We also need an uninitialized state  $\perp$  and initialization transitions.

# Characterizing eager invariance : LTS

Set of states  $\mathcal{S}$

$$\mathcal{S} = (\text{Config} \cup \{\perp\})^3$$

Set of actions  $Act$

$$Act = \{\tau\} \cup \{(a)_j \mid a \in \text{Action} \cup \{\text{init}\} \wedge j \in \{1, 2, 3\}\}$$

Labelled transition relation  $\rightarrow$

$$\frac{\cdot}{(\perp, s_2, s_3) \xrightarrow{(init)_1} (\langle S, \mu \rangle, s_2, s_3)} \quad \text{for all } \mu \in \text{Store}$$
$$\frac{c_1 \xrightarrow{\tau} c'_1}{(c_1, s_2, s_3) \xrightarrow{\tau} (c'_1, s_2, s_3)}$$
$$\frac{c_1 \xrightarrow{a} c'_1}{(c_1, s_2, s_3) \xrightarrow{(a)_1} (c'_1, s_2, s_3)}$$

# Characterizing eager invariance : formula

$\mu$ -calculus characterization (without initialization step)

$$\nu Y. \text{mimic}_{3,1} \wedge$$
$$\bigwedge_{a_h \in \text{H-Actions}} \llbracket (a_h)_1 \rrbracket Y$$
$$\bigwedge_{a_h \in \text{H-Actions}} \llbracket (a_h)_2 \rrbracket \langle\langle (a_h)_3 \rangle\rangle Y$$
$$\bigwedge_{a_l \in \text{L-Actions}} \llbracket (a_l)_1 \rrbracket \llbracket (a_l)_2 \rrbracket \langle\langle (a_l)_3 \rangle\rangle Y$$

and  $\text{mimic}_{3,1}$  is

$$\nu Z. \bigwedge_{a_l \in \text{L-Actions}} \llbracket (a_l)_1 \rrbracket_H \langle\langle (a_l)_3 \rangle\rangle_H Z$$

# Outline

## Comparing different proposals of Non-Interference

Uniform program model

Re-expressing proposals over our model

Comparison of the proposals

## Enforcing Non-Interference by temporal logic

Security type system vs model checking

Modal  $\mu$ -calculus

Characterizing Non-Interference properties in modal  $\mu$ -calculus

Model checking with the Concurrency Workbench (CWB)

# Model checking with the CWB

## Main features

- ▶ supports full modal  $\mu$ -calculus
- ▶ processes defined in CCS
- ▶ no data language (thus no parameterized actions, no conditional statements)

## Restrictions over our model (for simplicity)

- ▶ boolean variables only
- ▶ boolean expressions : constants or boolean names
- ▶ no (dead1) or (term) transition

# Model checking with the CWB : results

## Observational determinism

We have model checked all the examples that we had used to compare the different proposals. They are correctly accepted or rejected.

## Eager invariance

We could not have CWB to model check this property (state space explosion issue). It takes one fixed point operator to hide H transitions. Thus modalities that abstract away from H actions increase the nesting depth of the whole formula.

# Conclusion

## Unification of the different proposals

The definitions we have considered are uncomparable.

## About our experiments

- ▶ A characterization in modal *mu*-calculus is a good approach to achieve model checking of Non-Interference.
- ▶ We have not done any experiment with probabilistic properties.

## Alternative approach

Model checking with path equivalences leads to simpler characterizations.