

# Non-Interference for Multi-Agent Workflows

Helmut Seidl

Joint work with:

Christian Müller, Bernd Finkbeiner

TU München + U. des Saarlandes

MOD 2019



+



# Leader Election

```
while (*) {  
    choose  Msg( $m, y$ ) := Msg( $m, y$ )  $\vee$  Next( $m, y$ );  
    or {  
        Msg( $m, x$ ) := Msg( $m, x$ )  $\vee$   $\exists y$ .  
        Next( $y, x$ )  $\wedge$  Msg( $m, y$ )  $\wedge$  Less( $y, m$ );  
        Leader( $x$ ) := Leader( $x$ )  $\vee$  Msg( $x, x$ );  
    }  
}
```

// assuming appropriate axiomatization  
// for Next, Less

# Leader Election

```
while (*) {  
  choose   $\text{Msg}(m, y) := \text{Msg}(m, y) \vee \text{Next}(m, y);$   
  or {  
     $\text{Msg}(m, x) := \text{Msg}(m, x) \vee \exists y.$   
     $\text{Next}(y, x) \wedge \text{Msg}(m, y) \wedge \text{Less}(y, m);$   
     $\text{Leader}(x) := \text{Leader}(x) \vee \text{Msg}(x, x);$   
  }  
}
```

// assuming appropriate axiomatization

// for **Next**, **Less**

$\implies$  parametric system

# Conference Management

HotCRP.com

# Conference Management

HotCRP.com

## Multiple Agents

- PC chair
- authors
- reviewers
- ...

### Multiple Agents

- PC chair
- authors
- reviewers
- ...

**2.60** 19.Jul.2013

Major new feature: Paper managers. Administrators can assign PC members to "manage" individual papers. These PC members gain admin rights over those papers, and can, for example, assign reviewers as usual.

This required extensive rearchitecting of system internals to (hopefully) avoid information leaks.

# Conference Management

HotCRP.com

## Multiple Agents

- PC chair
- authors
- reviewers
- ...

2.60 19.Jul.2013

Major new feature: Paper managers. Administrators can assign PC members to "manage" individual papers. These PC members gain admin rights over those papers, and can, for example, assign reviewers as usual.

This required extensive rearchitcting of system internals to (hopefully) avoid information leaks.

Is sensitive information kept secret?

### Multiple Agents

- PC chair
- authors
- reviewers
- ...

2.60 19.Jul.2013

Major new feature: Paper managers. Administrators can assign PC members to "manage" individual papers. These PC members gain admin rights over those papers, and can, for example, assign reviewers as usual.

This required extensive rearchitecting of system internals to (hopefully) avoid information leaks.

Is sensitive information kept secret?  
How to protect sensitive information?

# Conference Management

Correctness, non-interference should be independent of

- ▶ the number of papers
- ▶ the number of authors
- ▶ the number of pc members

# Conference Management

Correctness, non-interference should be independent of

- ▶ the number of papers
- ▶ the number of authors
- ▶ the number of pc members

⇒ parametric system

# Parametric Workflow

# Parametric Workflow

Bernd Finkbeiner,



# Parametric Workflow

Bernd Finkbeiner, Chris Müller,



# Parametric Workflow

Bernd Finkbeiner, Chris Müller, H.S.  
2016, 2017, 2018, 2019



# Workflow

$\text{Conflict}(x, p) := \text{Author}(x, p) \vee A_2(x, p);$   
//  $A_2$  input predicate

# Workflow

$\text{Conflict}(x, p) := \text{Author}(x, p) \vee A_2(x, p);$

//  $A_2$  input predicate

$\text{Assign}(x, p) := \neg \text{Conflict}(x, p);$

// strategy of pc chair

# Workflow

$\text{Conflict}(x, p) := \text{Author}(x, p) \vee A_2(x, p);$

//  $A_2$  input predicate

$\text{Assign}(x, p) := \neg \text{Conflict}(x, p);$

// strategy of pc chair

**assume**  $\forall p. \exists x_1, x_2. x_1 \neq x_2 \wedge$   
 $\text{Assign}(x_1, p) \wedge \text{Assign}(x_2, p)$

# Workflow

$\text{Conflict}(x, p) := \text{Author}(x, p) \vee A_2(x, p);$   
//  $A_2$  input predicate

$\text{Assign}(x, p) := \neg \text{Conflict}(x, p);$   
// strategy of pc chair

**assume**  $\forall p. \exists x_1, x_2. x_1 \neq x_2 \wedge$   
 $\text{Assign}(x_1, p) \wedge \text{Assign}(x_2, p)$

$\text{Report}(x, p, r) := \text{Assign}(x, p) \wedge A_3(x, p, r)$

# Workflow

$\text{Conflict}(x, p) := \text{Author}(x, p) \vee A_2(x, p);$   
//  $A_2$  input predicate

$\text{Assign}(x, p) := \neg \text{Conflict}(x, p);$   
// strategy of pc chair

**assume**  $\forall p. \exists x_1, x_2. x_1 \neq x_2 \wedge$   
 $\text{Assign}(x_1, p) \wedge \text{Assign}(x_2, p)$

$\text{Report}(x, p, r) := \text{Assign}(x, p) \wedge A_3(x, p, r)$

$\text{Discuss}(x_1, x_2, p, d) := \exists r_1 r_2.$   
 $\text{Report}(x_1, p, r_1) \wedge$   
 $\text{Report}(x_2, p, r_2) \wedge$   
 $A_4(x_2, x_1, p, d)$

# Workflow

- ... uses predicates  $A$  for input from the environment or agent decisions
- ... offers **assume** statements

# Workflow

- ... uses predicates  $A$  for input from the environment or agent decisions
- ... offers **assume** statements
- ... applied to **non-interference**

# Plan of Tutorial

Lecture 1: Setting the Ground

Lecture 2: NI as an Invariant

Lecture 3: NI via Abstraction

Lecture 4: Playing with NI

# Lecture 1:

## Setting the Ground

# FO Transition System

Universe

$$U = \{\text{Gilles, Mooly, Bernd, Javier, \dots}\}$$

# FO Transition System

Universe

$$U = \{\text{Gilles, Mooly, Bernd, Javier, \dots}\}$$

Valuation  $\rho : \text{Constants} \rightarrow U$

# FO Transition System

Universe

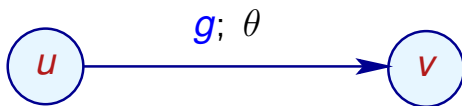
$$U = \{\text{Gilles, Mooly, Bernd, Javier, \dots}\}$$

Valuation  $\rho : \text{Constants} \rightarrow U$

State  $\equiv$  FO structure

$$\text{Assign} = \{ (\text{Gilles}, 17), (\text{Gilles}, 42), \\ (\text{Mooly}, 2), (\text{Mooly}, 17), \\ (\text{Bernd}, 42), (\text{Bernd}, 101), \\ \dots \}$$

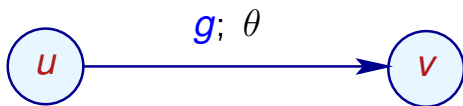
# Transition



$g$  condition

$\theta$  defines predicates at  $v$  in terms of state at  $u$  and input predicates.

# Transition

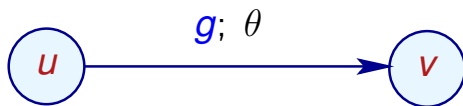


$g$  condition

$\theta$  defines predicates at  $v$  in terms of state at  $u$  and input predicates.

$$g = \forall p. \exists x. \text{Assign}(x, p)$$

# Transition



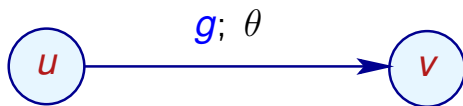
$g$  condition

$\theta$  defines predicates at  $v$  in terms of state at  $u$  and input predicates.

$$g = \forall p. \exists x. \text{Assign}(x, p)$$

$$\theta = \left\{ \begin{array}{ll} \text{Report}(x, p, r) & \mapsto \text{Assign}(x, p) \wedge A_2(x, p, r), \\ \text{Assign}(x, p) & \mapsto \text{Assign}(x, p), \\ \text{Conflict}(x, p) & \mapsto \text{Conflict}(x, p), \\ \text{Author}(x, p) & \mapsto \text{Author}(x, p) \end{array} \right\}$$

# Transition



$g$  condition

$\theta$  defines predicates at  $v$  in terms of state at  $u$  and input predicates.

$$g = \forall p. \exists x. \text{Assign}(x, p)$$

$$\theta = \left\{ \begin{array}{ll} \text{Report}(x, p, r) & := \text{Assign}(x, p) \wedge A_2(x, p, r), \\ \text{Assign}(x, p) & := \text{Assign}(x, p), \\ \text{Conflict}(x, p) & := \text{Conflict}(x, p), \\ \text{Author}(x, p) & := \text{Author}(x, p) \end{array} \right\}$$

# Initial Assumption

initial point —  $v_0$

# Initial Assumption

initial point —  $v_0$

initial state — Init

# Initial Assumption

initial point —  $v_0$

initial state —  $\text{Init}$

E.g.,

$$\forall x, p, r. \neg \text{Read}(x, p, r)$$

# Parametric NI

## Assumptions

Read( $x, p, r$ ) —

# Parametric NI

## Assumptions

$\text{Read}(x, p, r) \dashv x$  learns  $(x, p, r) \in \text{Read}$

# Parametric NI

## Assumptions

Read( $x, p, r$ ) —  $x$  learns  $(x, p, r) \in \text{Read}$   
secret-independent control

# Parametric NI

## Assumptions

Read( $x, p, r$ )  $\dashv x$  learns  $(x, p, r) \in \text{Read}$   
secret-independent control

## Approach

- ▶ Synchronous self-composition

# Parametric NI

## Assumptions

Read( $x, p, r$ ) —  $x$  learns  $(x, p, r) \in \text{Read}$   
secret-independent control

## Approach

- ▶ Synchronous self-composition
- ▶ Single out arbitrary agent  $a$
- ▶ Consider secret information relative to  $a$

# NI for Single Agent $a$

## First Attempt

$$NI(a) := \mathbf{G} \text{same\_observations}(a)$$

# NI for Single Agent $a$

## First Attempt

$$NI(a) := \mathbf{G} \text{same\_observations}(a)$$

$$\text{same\_observations}(a) := \bigwedge_R (\forall z. R a z \leftrightarrow R' a z)$$

# Declassification

## Issue

Information secret to others, could be disclosed to *a ...*

# Declassification

## Issue

Information secret to others, could be disclosed to  $a \dots$

$$NI(a) = \mathbf{G} \text{ same\_high\_inputs}(a) \rightarrow \mathbf{G} \text{ same\_observations}(a)$$

# Declassification

## Issue

Information secret to others, could be disclosed to  $a \dots$

$$NI(a) = \mathbf{G} \text{same\_high\_inputs}(a) \rightarrow \\ \mathbf{G} \text{same\_observations}(a)$$

$$\text{same\_high\_inputs}(a) = \bigwedge_I (\forall z. \text{declass}_I(a, z) \wedge \\ \text{declass}_{I'}(a, z) \rightarrow \\ (Iz \leftrightarrow I'z))$$

# Declassification

## Issue

Information secret to others, could be disclosed to  $a \dots$

$$NI(a) = \mathbf{G} \text{same\_high\_inputs}(a) \rightarrow \\ \mathbf{G} \text{same\_observations}(a)$$

$$\text{same\_high\_inputs}(a) = \bigwedge_l (\forall z. \text{declass}_l(a, z) \wedge \\ \text{declass}_{l'}(a, z) \rightarrow \\ (lz \leftrightarrow l'z))$$

E.g.,

$$\text{declass}_l(a, xpr) = \neg \text{Conflict}(a, p) \vee (x \neq a)$$

# Agent Model

## Issue 2

Adversaries may choose different  $A$  on different tracks

# Agent Model

## Issue 2

Adversaries may choose different  $A$  on different tracks

$$\text{Conflict}(x, p) \quad := \quad A(x, p) \vee \text{Author}(x, p)$$

$$\text{Conflict}'(x, p) \quad := \quad A'(x, p) \vee \text{Author}'(x, p)$$

# Agent Model (cont.)

Agent  $x$  could be

**Stubborn** Choices are independent of secrets:

$$\text{stubborn}(x) = G \text{ same\_low\_inputs}(x)$$

# Agent Model (cont.)

Agent  $x$  could be

**Stubborn** Choices are independent of secrets:

$$\text{stubborn}(x) = G \text{ same\_low\_inputs}(x)$$

$$\text{same\_low\_inputs}(x) = \bigwedge_A \forall z. Axz \leftrightarrow A'xz$$

# Agent Model (cont.)

Agent  $x$  could be

**Stubborn** Choices are **independent** of secrets:

$$\text{stubborn}(x) = G \text{ same\_low\_inputs}(x)$$

$$\text{same\_low\_inputs}(x) = \bigwedge_A \forall z. Axz \leftrightarrow A'xz$$

**Causal** Choices depend on **informedness** on secrets:

$$\begin{aligned} \text{causal}(x) = & \text{same\_low\_inputs}(x) \\ & W \\ & \neg \text{same\_observations}(x) \end{aligned}$$

# Summary

## Parametric NI with declassification and agent model

$$\begin{aligned} &(\forall x. \text{stubborn}(x)) \rightarrow \\ &\quad (\mathbf{G} \text{ same\_high\_inputs}(a)) \rightarrow \\ &\quad\quad (\mathbf{G} \text{ same\_observations}(a)) \end{aligned}$$

# Summary

## Parametric NI with declassification and agent model

$$\begin{aligned} (\forall x. stubborn(x)) \rightarrow \\ (\mathbf{G} \text{ same\_high\_inputs}(a)) \rightarrow \\ (\mathbf{G} \text{ same\_observations}(a)) \end{aligned}$$

$$\begin{aligned} (\forall x. causal(x)) \rightarrow \\ (\mathbf{G} \text{ same\_high\_inputs}(a)) \rightarrow \\ (\mathbf{G} \text{ same\_observations}(a)) \end{aligned}$$

# Summary

## Parametric NI with declassification and agent model

$$(\forall x. stubborn(x)) \rightarrow \\ (\mathbf{G} \text{ same\_high\_inputs}(a)) \rightarrow \\ (\mathbf{G} \text{ same\_observations}(a))$$

$$(\forall x. causal(x)) \rightarrow \\ (\mathbf{G} \text{ same\_high\_inputs}(a)) \rightarrow \\ (\mathbf{G} \text{ same\_observations}(a))$$

FOLTL

# Code Everything into FOLTL

# Code Everything into FOLTL

- ▶ Introduce nullary predicates  $\nu$ ,  $\nu$  program point

# Code Everything into FOLTL

- ▶ Introduce nullary predicates  $\psi$ ,  $\psi$  program point
- ▶ For each  $\psi$ , introduce

$$\psi_{\psi} = \neg \psi \vee \bigvee_{(v, g?, v') \text{ edge}} g \wedge \chi_{\psi'}$$

# Code Everything into FOLTL

- ▶ Introduce nullary predicates  $v$ ,  $v$  program point
- ▶ For each  $v$ , introduce

$$\psi_v = \neg v \vee \bigvee_{(v, g? \cdot, v') \text{ edge}} g \wedge \mathbf{X} v'$$

- ▶ For edge  $e = (v, g? \theta, v')$ , introduce

$$\begin{aligned} \phi_e &= \neg v \vee \neg g \vee \neg \mathbf{X} v' \vee \\ &\quad (\bigwedge_R \forall z. \mathbf{X} Rz \leftrightarrow (Rz)\theta) \end{aligned}$$

# Code Everything into FOLTL

- ▶ Introduce nullary predicates  $v$ ,  $v$  program point
- ▶ For each  $v$ , introduce

$$\psi_v = \neg v \vee \bigvee_{(v, g? \cdot, v') \text{ edge}} g \wedge \mathbf{X} v'$$

- ▶ For edge  $e = (v, g? \theta, v')$ , introduce

$$\begin{aligned} \phi_e = \neg v \vee \neg g \vee \neg \mathbf{X} v' \vee \\ (\bigwedge_R \forall z. \mathbf{X} Rz \leftrightarrow (Rz)\theta) \end{aligned}$$

- ▶ Encode FOTS + spec into FOLTL:

$$v_0 \wedge \text{Init} \wedge \left( \mathbf{G} \bigwedge_v \psi_v \wedge \bigwedge_e \phi_e \right) \rightarrow \forall a. \text{PNID}(a)$$

# Theorem

Falsification of parametric NID for FOTS can be reduced to satisfiability of FOLTL

# Theorem

Falsification of parametric NID for FOTS can be reduced to satisfiability of FOLTL

FOLTL is undecidable (no wonder)

# Theorem

Falsification of parametric NID for FOTS can be reduced to satisfiability of FOLTL

FOLTL is undecidable (no wonder)

decidable fragments ??

# Quantifierfree FOTS

$\text{Conflict}(x, p) := \neg \text{Author}(x, p) \wedge A_2(x, p);$

//  $A$  input predicate

$\text{Assign}(x, p) := \neg \text{Conflict}(x, p);$

// strategy of pc chair

$\text{Report}(x, p, r) := \text{Assign}(x, p) \wedge I(x, p, r);$

$\text{Discuss}(x_1, x_2, p, r_1, r_2, d) := \text{Report}(x_1, p, r_1) \wedge$   
 $\text{Report}(x_2, p, r_2) \wedge$   
 $A_4(x_2, x_1, p, d)$

# Quantifierfree FOTS

$\text{Conflict}(x, p) := \neg \text{Author}(x, p) \wedge A_2(x, p);$

//  $A$  input predicate

$\text{Assign}(x, p) := \neg \text{Conflict}(x, p);$

// strategy of pc chair

$\text{Report}(x, p, r) := \text{Assign}(x, p) \wedge I(x, p, r);$

$\text{Discuss}(x_1, x_2, p, r_1, r_2, d) := \text{Report}(x_1, p, r_1) \wedge$   
 $\text{Report}(x_2, p, r_2) \wedge$   
 $A_4(x_2, x_1, p, d)$



quantifierfree!!

# Quantifierfree FOTS

Init in  $\exists^*\forall^*$  FOL

$\implies$  **Stubborn**  $\neg$ PNID translates into  $\exists^*\forall^*$  **FOLTL**.

# Quantifierfree FOTS

Init in  $\exists^*\forall^*$  FOL

- $\implies$  **Stubborn**  $\neg$ PNID translates into  $\exists^*\forall^*$  **FOLTL**.
- $\implies$  **k-Causal**  $\neg$ PNID translates into  $\exists^*\forall^*$  **FOLTL**.

# Quantifierfree FOTS

Init in  $\exists^*\forall^*$  FOL

$\implies$  **Stubborn**  $\neg$ PNID translates into  $\exists^*\forall^*$  FOLTL.

$\implies$  **k-Causal**  $\neg$ PNID translates into  $\exists^*\forall^*$  FOLTL.

## Theorem

Kuperberg, Brunel, Chemouil 2016

Satisfiability of  $\exists^*\forall^*$  FOLTL is decidable.

# Quantifierfree FOTS

Init in  $\exists^*\forall^*$  FOL

$\implies$  **Stubborn**  $\neg$ PNID translates into  $\exists^*\forall^*$ FOLTL.

$\implies$  **k-Causal**  $\neg$ PNID translates into  $\exists^*\forall^*$ FOLTL.

## Theorem

Kuperberg, Brunel, Chemouil 2016

Satisfiability of  $\exists^*\forall^*$ FOLTL is decidable.

## Corollary

**Stubborn** / **k-Causal** PNID is decidable.

# Results for HotCRP

# Results for HotCRP

Stubborn PNID holds!

# Results for HotCRP

Causal PNID fails ...

# Results for HotCRP

Causal PNID fails ...

Counter Example

$$\text{Conflict} = \{(a_1, p_1)\}$$

# Results for HotCRP

Causal PNID fails ...

## Counter Example

Conflict =  $\{(a_1, p_1)\}$

Assign =  $\{(a_2, p_1), (a_1, p_2), (a_2, p_2)\}$

Report =  $\{(a_2, p_1, r),$   
 $(a_1, p_2, r_1), (a_2, p_2, r_2)\}$

# Results for HotCRP

Causal PNID fails ...

## Counter Example

Conflict =  $\{(a_1, p_1)\}$

Assign =  $\{(a_2, p_1), (a_1, p_2), (a_2, p_2)\}$

Report =  $\{(a_2, p_1, r),$   
 $(a_1, p_2, r_1), (a_2, p_2, r_2)\}$

Report' =  $\{(a_2, p_1, r'),$   
 $(a_1, p_2, r_1), (a_2, p_2, r_2)\}$

# Results for HotCRP

Causal PNID fails ...

## Counter Example

Conflict =  $\{(a_1, p_1)\}$

Assign =  $\{(a_2, p_1), (a_1, p_2), (a_2, p_2)\}$

Report =  $\{(a_2, p_1, r),$   
 $(a_1, p_2, r_1), (a_2, p_2, r_2)\}$

Report' =  $\{(a_2, p_1, r'),$   
 $(a_1, p_2, r_1), (a_2, p_2, r_2)\}$

Discuss =  $\{(a_1, a_2, p_2, r_1, r_2, d)\}$

Discuss' =  $\{(a_1, a_2, p_2, r_1, r_2, d')\}$

# Summary

- ▶ FO transition systems are an expressive tool to specify parametric systems

# Summary

- ▶ FO transition systems are an expressive tool to specify parametric systems
- ▶ Parametric non-interference can be formalized by means of FOLTL

# Summary

- ▶ FO transition systems are an expressive tool to specify parametric systems
- ▶ Parametric non-interference can be formalized by means of FOLTL
- ▶ Violation of PNID reduces to satisfiability of FOLTL

# Summary

- ▶ FO transition systems are an expressive tool to specify parametric systems
- ▶ Parametric non-interference can be formalized by means of **FOLTL**
- ▶ Violation of PNID reduces to **satisfiability** of **FOLTL**
- ▶ PNID for quantifierfree FOTS is **decidable**.