

Non-Interference for Multi-Agent Workflows

Helmut Seidl

Joint work with:

Christian Müller, Bernd Finkbeiner

TU München + U. des Saarlandes

MOD 2019

Lecture 2:

Non-Interference as an Invariant

Recap

Parametric NI with declassification and agent model

Recap

Parametric NI with declassification and agent model

$$\begin{aligned} &(\forall x. stubborn(x)) \rightarrow \\ &\quad (\mathbf{G} \text{ same_high_inputs}(a)) \rightarrow \\ &\quad\quad (\mathbf{G} \text{ same_observations}(a)) \end{aligned}$$

Recap

Parametric NI with declassification and agent model

$$(\forall x. stubborn(x)) \rightarrow \\ (\mathbf{G} \text{ same_high_inputs}(a)) \rightarrow \\ (\mathbf{G} \text{ same_observations}(a))$$

$$(\forall x. causal(x)) \rightarrow \\ (\mathbf{G} \text{ same_high_inputs}(a)) \rightarrow \\ (\mathbf{G} \text{ same_observations}(a))$$

Recap

Parametric NI with declassification and agent model

$$(\forall x. \textit{stubborn}(x)) \rightarrow \\ (\mathbf{G} \textit{same_high_inputs}(a)) \rightarrow \\ (\mathbf{G} \textit{same_observations}(a))$$

$$(\forall x. \textit{causal}(x)) \rightarrow \\ (\mathbf{G} \textit{same_high_inputs}(a)) \rightarrow \\ (\mathbf{G} \textit{same_observations}(a))$$



expressive logic



works for restricted class of FOTS only !?

Alternative Approach

Simplify formula by complicating the FOTS!!

Alternative Approach

Simplify formula by complicating the FOTS!!

- ▶ Synchronous self-composition
- ▶ Encode declassification into FOTS
- ▶ Encode agent model into FOTS

Alternative Approach

Simplify formula by complicating the FOTS!!

- ▶ Synchronous self-composition
- ▶ Encode declassification into FOTS
- ▶ Encode agent model into FOTS

Result

Property to be verified

G same_observations(a)

Alternative Approach

Simplify formula by complicating the FOTS!!

- ▶ Synchronous self-composition
- ▶ Encode declassification into FOTS
- ▶ Encode agent model into FOTS

Result

Property to be verified

\mathbf{G} same_observations(a)

\implies Invariant

Synchronous Self-Composition

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

Synchronous Self-Composition

$$\left\{ \begin{array}{l} \text{Assign}(x, p) \quad := \quad \neg \text{Conflict}(x, p); \\ \text{Assign}'(x, p) \quad := \quad \neg \text{Conflict}'(x, p); \end{array} \right\}$$

Declassification

{

Report(x, p, r) := Assign(x, p) \wedge $I(x, p, r)$

Declassification

{

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

$\text{Report}'(x, p, r) := \text{Assign}'(x, p) \wedge$
 $((\neg \text{declass}_I(a, xpr) \vee$
 $\neg \text{declass}'_I(a, xpr)) \vee I(x, p, r)) \wedge$

Declassification

$$\{$$
$$\text{Report}(x, p, r) := \text{Assign}(x, p) \wedge I(x, p, r)$$
$$\text{Report}'(x, p, r) := \text{Assign}'(x, p) \wedge$$
$$((\neg \text{declass}_I(a, xpr) \vee$$
$$\neg \text{declass}'_I(a, xpr)) \vee I(x, p, r)) \wedge$$
$$((\text{declass}_I(a, xpr) \wedge$$
$$\text{declass}'_I(a, xpr)) \vee I'(x, p, r))$$
$$\}$$

Stubborn Agents

use the same A on both tracks

Stubborn Agents

use the same A on both tracks

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

Stubborn Agents

use the same A on both tracks

$$\left\{ \begin{array}{l} \text{Discuss}(x_1, x_2, p, r_1, r_2, d) := \text{Report}(x_1, p, r_1) \wedge \\ \text{Report}(x_2, p, r_2) \wedge \\ A_4(x_2, x_1, p, d) \\ \text{Discuss}'(x_1, x_2, p, r_1, r_2, d) := \text{Report}'(x_1, p, r_1) \wedge \\ \text{Report}'(x_2, p, r_2) \wedge \\ A_4(x_2, x_1, p, d) \end{array} \right\}$$

Causal Agents

Auxiliary predicate **Informed**

Causal Agents

Auxiliary predicate **Informed**

$$\text{Informed}(x) := \text{Informed}(x) \vee \\ \bigvee_R \exists z. Rxz \not\leftrightarrow R'xz$$

Causal Agents

Auxiliary predicate **Informed**

$$\begin{aligned} \text{Informed}(x) &:= \text{Informed}(x) \vee \\ &\quad \bigvee_R \exists z. Rxz \not\leftrightarrow R'xz \end{aligned}$$

keeps track of all agents who know about secrets ...

Causal Agents (cont.)

... can be used to further propagate secrets

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

Causal Agents (cont.)

... can be used to further propagate secrets

$$\left\{ \begin{array}{l} \text{Discuss}(x_1, x_2, p, r_1, r_2, d) := \text{Report}(x_1, p, r_1) \wedge \\ \text{Report}(x_2, p, r_2) \wedge \\ A_4(x_2, x_1, p, d) \\ \text{Discuss}'(x_1, x_2, p, r_1, r_2, d) := \text{Report}'(x_1, p, r_1) \wedge \\ \text{Report}'(x_2, p, r_2) \wedge \\ (\text{Informed}(x_2) \vee A_4(x_2, x_1, p, d)) \wedge \\ (\neg \text{Informed}(x_2) \vee A'_4(x_2, x_1, p, d)) \end{array} \right\}$$

Invariants for FOTS

Given invariant I
initial condition $Init$ for v_0

Invariants for FOTS

Given invariant I
initial condition Init for v_0

Compute

$$\Psi^{(0)}[u] = I[u]$$

$$\Psi^{(h)}[u] = \Psi^{(h-1)}[u] \wedge \bigwedge_{(u,g?\theta,v) \in E} \neg g \vee \forall A. (\Psi^{(h-1)}[v]\theta)$$

// weakest precondition !

for $h > 0$

Invariants for FOTS

Given invariant I
initial condition Init for v_0

Compute

$$\Psi^{(0)}[u] = I[u]$$

$$\Psi^{(h)}[u] = \Psi^{(h-1)}[u] \wedge \bigwedge_{(u,g?\theta,v) \in E} \neg g \vee \forall A. (\Psi^{(h-1)}[v]\theta)$$

// weakest precondition !

for $h > 0$

Then invariant I holds iff

$$\text{Init} \Rightarrow \Psi^{(h)}[v_0] \quad (h \geq 0)$$

Simplified HotCRP

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

// A_2 input predicate

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

// strategy of pc chair

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

Simplified HotCRP

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

// A_2 input predicate

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

// strategy of pc chair

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

Invariant

$I[v_3] = \forall x, p, r. \neg \text{Author}(x, p) \vee \neg \text{Report}(x, p, r)$

Iteration

$$\psi^{(0)}[v_3] = \forall x, p, r. \neg \text{Author}(x, p) \vee \neg \text{Report}(x, p, r)$$

Iteration

$$\psi^{(0)}[v_3] = \forall x, p, r. \neg \text{Author}(x, p) \vee \neg \text{Report}(x, p, r)$$

$$\begin{aligned} \psi^{(1)}[v_2] &= \forall A_3, x, p, r. \neg \text{Author}(x, p) \vee \neg \text{Assign}(x, p) \vee \neg A_3(x, p, r) \\ &= \forall x, p. \neg \text{Author}(x, p) \vee \neg \text{Assign}(x, p) \end{aligned}$$

Iteration

$$\psi^{(0)}[v_3] = \forall x, p, r. \neg \text{Author}(x, p) \vee \neg \text{Report}(x, p, r)$$

$$\begin{aligned} \psi^{(1)}[v_2] &= \forall A_3, x, p, r. \neg \text{Author}(x, p) \vee \neg \text{Assign}(x, p) \vee \neg A_3(x, p, r) \\ &= \forall x, p. \neg \text{Author}(x, p) \vee \neg \text{Assign}(x, p) \end{aligned}$$

$$\psi^{(2)}[v_1] = \forall x, p. \neg \text{Author}(x, p) \vee \text{Conflict}(x, p)$$

Iteration

$$\psi^{(0)}[v_3] = \forall x, p, r. \neg \text{Author}(x, p) \vee \neg \text{Report}(x, p, r)$$

$$\begin{aligned}\psi^{(1)}[v_2] &= \forall A_3, x, p, r. \neg \text{Author}(x, p) \vee \neg \text{Assign}(x, p) \vee \neg A_3(x, p, r) \\ &= \forall x, p. \neg \text{Author}(x, p) \vee \neg \text{Assign}(x, p)\end{aligned}$$

$$\psi^{(2)}[v_1] = \forall x, p. \neg \text{Author}(x, p) \vee \text{Conflict}(x, p)$$

$$\begin{aligned}\psi^{(3)}[v_0] &= \forall x, p. \neg \text{Author}(x, p) \vee \text{Author}(x, p) \\ &= \top\end{aligned}$$

Iteration

$$\psi^{(0)}[v_3] = \forall x, p, r. \neg \text{Author}(x, p) \vee \neg \text{Report}(x, p, r)$$

$$\begin{aligned}\psi^{(1)}[v_2] &= \forall A_3, x, p, r. \neg \text{Author}(x, p) \vee \neg \text{Assign}(x, p) \vee \neg A_3(x, p, r) \\ &= \forall x, p. \neg \text{Author}(x, p) \vee \neg \text{Assign}(x, p)\end{aligned}$$

$$\psi^{(2)}[v_1] = \forall x, p. \neg \text{Author}(x, p) \vee \text{Conflict}(x, p)$$

$$\begin{aligned}\psi^{(3)}[v_0] &= \forall x, p. \neg \text{Author}(x, p) \vee \text{Author}(x, p) \\ &= \top\end{aligned}$$

What about loops ??

Inductive Invariants

Ψ proves / if

Inductive Invariants

Ψ proves I if

▶ $\Psi[u] \rightarrow I[u]$ for all u // strengthening

Inductive Invariants

Ψ proves I if

▶ $\Psi[u] \rightarrow I[u]$ for all u // strengthening

▶ $\Psi[u] \rightarrow \neg g \vee \Psi[v]\theta$ // inductivity

Inductive Invariants

Ψ proves I if

- ▶ $\Psi[u] \rightarrow I[u]$ for all u // strengthening
- ▶ $\Psi[u] \rightarrow \neg g \vee \Psi[v]\theta$ // inductivity
- ▶ $\text{Init} \rightarrow \Psi[v_0]$ // validity

Inductive Invariants

Ψ proves I if

- ▶ $\Psi[u] \rightarrow I[u]$ for all u // strengthening
- ▶ $\Psi[u] \rightarrow \neg g \vee \Psi[v]\theta$ // inductivity
- ▶ $\text{Init} \rightarrow \Psi[v_0]$ // validity

Obstacles

- ▶ The required invariant is not expressible in **FOL**
- ▶ The required implications are too complicated
- ▶ There is a simple invariant, but we have no idea

EPR

Given

ψ, l in \forall^* FOL

$init, g$ in $\exists^*\forall^*$ FOL

$(Rz)\theta$ in $(\forall^* \cup \exists^*)$ FOL

EPR

Given

Ψ, I in \forall^* FOL

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

$(Rz)\theta$ in $(\forall^* \cup \exists^*)$ FOL

Then

I can be effectively checked via Ψ

EPR

Given

Ψ, I in \forall^* FOL

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

$(Rz)\theta$ in $(\forall^* \cup \exists^*)$ FOL

Then

I can be effectively checked via Ψ

\implies applicable to many more programs

EPR

Given

Ψ, I in \forall^* FOL

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

$(Rz)\theta$ in $(\forall^* \cup \exists^*)$ FOL

Then

I can be effectively checked via Ψ

- \implies applicable to many more programs
- \implies semi-automatic only

Application to PNID

$R\theta$ in $(\forall^* \cup \exists^*)$ FOL

no guards



preserved by synchronous self-composition

Application to PNID

$R\theta$ in $(\forall^* \cup \exists^*)$ FOL

no guards



preserved by synchronous self-composition

Corollary

PNID can be effectively verified given an inductive strengthening in \forall^* FOL.

Application to PNID

$R\theta$ in $(\forall^* \cup \exists^*)$ FOL

no guards



preserved by synchronous self-composition

Corollary

PNID can be effectively verified given an inductive strengthening in \forall^* FOL.

... how to find valid inductive strengthenings??

Conceiving Inductive Invariants

Issues to be solved

- ▶ FO implication checking
- ▶ Termination of fixpoint iteration

Conceiving Inductive Invariants

Issues to be solved

- ▶ FO implication checking
- ▶ Termination of fixpoint iteration
- ▶ SO quantifier elimination

Monadic Predicate Logic

- ... is decidable
- ... admits SO quantifier elimination
- ... can be used to **abstract** general FOTS

Monadic Predicate Logic

- ... is decidable
- ... admits SO quantifier elimination
- ... can be used to **abstract** general FOTS

Theorem

Igor Walukiewicz

Safety for Monadic FOTS is **undecidable**.

Monadic Predicate Logic

- ... is decidable
- ... admits SO quantifier elimination
- ... can be used to **abstract** general FOTS

Theorem

Igor Walukiewicz

Safety for Monadic FOTS is **undecidable**.

Idea

- ▶ simulate counter by means of unary predicate
- ▶ use A , \neq to single out elements to add/remove

Counter Operations

Increment

$$\begin{aligned}\{P(x) &:= P(x) \vee A(x) \\ \text{err} &:= \text{err} \vee (\forall y. \neg A(y)) \vee\end{aligned}$$

Counter Operations

Increment

$$\begin{aligned}\{P(x) &:= P(x) \vee A(x) \\ \text{err} &:= \text{err} \vee (\forall y. \neg A(y)) \vee \\ &\quad (\exists y. A(y) \wedge P(y)) \vee\end{aligned}$$

Counter Operations

Increment

$$\begin{aligned} \{P(x) &:= P(x) \vee A(x) \\ \text{err} &:= \text{err} \vee (\forall y. \neg A(y)) \vee \\ &\quad (\exists y. A(y) \wedge P(y)) \vee \\ &\quad (\exists y_1, y_2. y_1 \neq y_2 \wedge A(y_1) \wedge A(y_2)) \\ \} \end{aligned}$$

Counter Operations

Increment

$$\begin{aligned} \{P(x) &:= P(x) \vee A(x) \\ \text{err} &:= \text{err} \vee (\forall y. \neg A(y)) \vee \\ &\quad (\exists y. A(y) \wedge P(y)) \vee \\ &\quad (\exists y_1, y_2. y_1 \neq y_2 \wedge A(y_1) \wedge A(y_2)) \\ \} \end{aligned}$$

Decrement

$$\begin{aligned} \{P(x) &:= P(x) \wedge \neg A(x) \\ \text{err} &:= \text{err} \vee (\forall y. \neg A(y)) \vee \end{aligned}$$

Counter Operations

Increment

$$\begin{aligned} \{P(x) &:= P(x) \vee A(x) \\ \text{err} &:= \text{err} \vee (\forall y. \neg A(y)) \vee \\ &\quad (\exists y. A(y) \wedge P(y)) \vee \\ &\quad (\exists y_1, y_2. y_1 \neq y_2 \wedge A(y_1) \wedge A(y_2)) \\ \} \end{aligned}$$

Decrement

$$\begin{aligned} \{P(x) &:= P(x) \wedge \neg A(x) \\ \text{err} &:= \text{err} \vee (\forall y. \neg A(y)) \vee \\ &\quad (\exists y. A(y) \wedge \neg P(y)) \vee \end{aligned}$$

Counter Operations

Increment

$$\begin{aligned} \{P(x) &:= P(x) \vee A(x) \\ \text{err} &:= \text{err} \vee (\forall y. \neg A(y)) \vee \\ &\quad (\exists y. A(y) \wedge P(y)) \vee \\ &\quad (\exists y_1, y_2. y_1 \neq y_2 \wedge A(y_1) \wedge A(y_2)) \\ \} \end{aligned}$$

Decrement

$$\begin{aligned} \{P(x) &:= P(x) \wedge \neg A(x) \\ \text{err} &:= \text{err} \vee (\forall y. \neg A(y)) \vee \\ &\quad (\exists y. A(y) \wedge \neg P(y)) \vee \\ &\quad (\exists y_1, y_2. y_1 \neq y_2 \wedge A(y_1) \wedge A(y_2)) \\ \} \end{aligned}$$

Initially

$$\mathit{Init} \equiv \dots \wedge \neg \mathit{err}$$

Initially

$$Init \equiv \dots \wedge \neg err$$

Invariant

$$I \equiv err \vee \neg final$$

... Decidable Subclasses

Theorem

Safety remains decidable when there is no A .

... Decidable Subclasses

Theorem

Safety remains decidable when there is no A .

Idea

Use variants of **Eliminations-hauptform** (**Behmann, 1921**), to deduce that quantifier nesting remains bounded.

... Decidable Subclasses

Theorem

Safety remains decidable when there is no A .

Idea

Use variants of **Eliminations-
hauptform** (**Behmann, 1921**), to
deduce that quantifier nesting
remains bounded.



... Decidable Subclasses (cont.)

Theorem

Safety remains decidable when there is A , but

- ▶ no “=, \neq ” in substitutions or `Init`.
- ▶ only “=” in invariant.

... Decidable Subclasses (cont.)

Theorem

Safety remains decidable when there is A , but

- ▶ no “=, \neq ” in substitutions or `Init`.
- ▶ only “=” in invariant.

Idea

QE of $\forall A$ introduces “=”

\implies compute **weakest strengthening** without “=”.
// best abstraction

Example

Consider the FOTS with single edge (u, θ, u) where

$$\theta = \begin{cases} Ry & := Ry \vee Ay \\ Sy & := Sy \vee Ay \vee \forall z. \neg Rz \end{cases}$$

Example

Consider the FOTS with single edge (u, θ, u) where

$$\theta = \begin{cases} Ry & := Ry \vee Ay \\ Sy & := Sy \vee Ay \vee \forall z. \neg Rz \end{cases}$$

Invariant

$$I[u] \equiv (\forall x. \neg Rx) \vee (\forall y. Sy)$$

Example (cont)

$$\psi^{(0)}[u] = (\forall x. \neg Rx) \vee (\forall y. Sy)$$

Example (cont)

$$\psi^{(0)}[u] = (\forall x. \neg Rx) \vee (\forall y. Sy)$$

$$\psi^{(1)}[u] = ((\forall x. \neg Rx) \vee (\forall y. Sy)) \wedge \\ \forall A. (\forall x, y, z. \neg Rx \wedge \neg Ax \vee Sy \vee Ay \vee \neg Rz)$$

Example (cont)

$$\psi^{(0)}[u] = (\forall x. \neg Rx) \vee (\forall y. Sy)$$

$$\begin{aligned} \psi^{(1)}[u] &= ((\forall x. \neg Rx) \vee (\forall y. Sy)) \wedge \\ &\quad \forall A. (\forall x, y, z. \neg Rx \wedge \neg Ax \vee Sy \vee Ay \vee \neg Rz) \end{aligned}$$

$$\begin{aligned} &= ((\forall x. \neg Rx) \vee (\forall y. Sy)) \wedge \\ &\quad (\forall x, y, z. \neg Rx \wedge (x = y) \vee Sy \vee \neg Rz) \end{aligned}$$

Example (cont)

$$\psi^{(0)}[u] = (\forall x. \neg Rx) \vee (\forall y. Sy)$$

$$\begin{aligned} \psi^{(1)}[u] &= ((\forall x. \neg Rx) \vee (\forall y. Sy)) \wedge \\ &\quad \forall A. (\forall x, y, z. \neg Rx \wedge \neg Ax \vee Sy \vee Ay \vee \neg Rz) \end{aligned}$$

$$\begin{aligned} &= ((\forall x. \neg Rx) \vee (\forall y. Sy)) \wedge \\ &\quad (\forall x, y, z. \neg Rx \wedge (x = y) \vee Sy \vee \neg Rz) \end{aligned}$$

$$= ((\forall x. \neg Rx) \vee (\forall y. Sy))$$

Equality Abstraction

$$\exists x. Px \wedge Qy \wedge (x = y) \quad \equiv \quad Py \wedge Qy$$

Equality Abstraction

$$\exists x. Px \wedge Qy \wedge (x = y) \quad \equiv \quad Py \wedge Qy$$

$$\forall z. Pz \wedge Qy \wedge (z = y) \vee F \quad \leftarrow \quad \forall z. F$$

Equality Abstraction

$$\exists x. Px \wedge Qy \wedge (x = y) \quad \equiv \quad Py \wedge Qy$$

$$\forall z. Pz \wedge Qy \wedge (z = y) \vee F \quad \leftarrow \quad \forall z. F$$

- ▶ Bring Ψ into **prenex NF**
- ▶ Bring quantifierfree part into DNF
- ▶ Apply rules

Equality Abstraction

$$\exists x. Px \wedge Qy \wedge (x = y) \quad \equiv \quad Py \wedge Qy$$

$$\forall z. Pz \wedge Qy \wedge (z = y) \vee F \quad \leftarrow \quad \forall z. F$$

- ▶ Bring Ψ into **prenex** NF
- ▶ Bring quantifierfree part into DNF
- ▶ Apply rules

The number of monadic **FOL** formulas is finite

\implies fixpoint iteration terminates!

Application to PNID

Synchronous self-composition does not introduce non-monadic auxiliary predicates or $=, \neq$.

Application to PNID

Synchronous self-composition does not introduce non-monadic auxiliary predicates or $=, \neq$.

Since the “=” abstraction is **best**, we obtain:

Corollary

PNID, both **stubborn** and **causal**, for monadic FOTS w/o $=, \neq$ is decidable.

Summary

- ▶ We simplified PNID to an **invariant** by coding self-composition, agent model and declassification into the FOTS.

Summary

- ▶ We simplified PNID to an **invariant** by coding self-composition, agent model and declassification into the FOTS.
- ▶ For a large class of FOTS, invariants can be **verified** by means of valid inductive strengthenings.

Summary

- ▶ We simplified PNID to an **invariant** by coding self-composition, agent model and declassification into the FOTS.
- ▶ For a large class of FOTS, invariants can be **verified** by means of valid inductive strengthenings.
- ▶ In order to solve the **inference** problem, SO quantifier elimination is required.

Summary

- ▶ We simplified PNID to an **invariant** by coding self-composition, agent model and declassification into the FOTS.
- ▶ For a large class of FOTS, invariants can be **verified** by means of valid inductive strengthenings.
- ▶ In order to solve the **inference** problem, SO quantifier elimination is required.
- ▶ For **monadic** FOTS decidability was obtained when all “=”, “≠” were absent.