A Process Algebra for (Delimited) Persistent Stochastic Non-Interference

Andrea Marin$^1$ Carla Piazza$^2$ Sabina Rossi$^1$

$^1$ Università Ca’ Foscari Venezia, Italy

$^2$ Università degli Studi di Udine, Italy

CILC 2019 - VALUETOOLS 2019 and QEST 2019
Persistent Stochastic Non-Interference

The Context

- **Non-Interference** aims at protecting sensitive data from undesired accesses.

- **Goguen-Meseguer'82**: information does not flow from high (confidential) to low (public) if the high behavior cannot be observed at low level.

- Few results deal with **time behaviour** and Non-Interference.

- **Persistency**: Non-Interference has to be guaranteed in all the states of the system, if processes migrate during execution.
Intuitively

Low level user observing the timing behaviour
**Motivation - I**

- **Non-Interference** could be **too demanding**. It does not allow any information flow

- **Delimited**: mechanisms for downgrading or declassifying information from **high** to **low** are necessary

- **Downgrading** of information has to be performed by a **trusted** component
Motivation - II

- Once a process has been designed, it is necessary to check whether it satisfies Delimited Non-Interference or not

- If the process is not secure, it is necessary to modify it

- We look for a language which defines only secure processes
Introduction

Delimited Persistent Stochastic Non-Interference

Contribution

- We introduce Persistent Stochastic Non-Interference (PSNI) Delimited Persistent Stochastic Non-Interference (D_PSNI) over Performance Evaluation Process Algebra (PEPA)
- We define process algebras for PSNI and D_PSNI processes
- Our process algebras denote equivalence relations that are
  - stronger than lumpability (bisimulation)
  - linearly verifiable w.r.t. the syntax of the process
OUTLINE OF THE TALK

► Performance Evaluation Process Algebra (PEPA)
► Observation Equivalence: Lumpable Bisimilarity
► Persistent Stochastic Non-Interference (PSNI)
► Delimited Persistent Stochastic Non-Interference (D_PSNI)
► Unwinding and Compositionality: two secure process algebras
► Example and Conclusions
PEPA - Syntax and Semantics

**Definition - PEPA Syntax**

Let $\mathcal{A}$ be a set of actions with $\tau \in \mathcal{A}$

Let $\alpha \in \mathcal{A}$, $\mathcal{A} \subseteq \mathcal{A}$, and $r \in \mathbb{R} \cup \{\top\}$

$$
S ::= 0 \mid (\alpha, r).S \mid S + S \mid X
$$

$$
P ::= P \boxplus \mathcal{A} P \mid P/A \mid P \setminus A \mid S
$$

Each variable $X$ is associated to a definition $X \equiv P$

**Definition - PEPA Semantics**

It defines Labeled Continuous Time Markov Chains
**Example**

\[
X_1 = (req_L, \rho).X_2 \\
X_2 = (res_L, \mu).X_1 + (log_H, \lambda).X_3 \\
X_3 = (res_L, \mu).X_1
\]
PEPA - Semantics for Synchronization

\[
\begin{align*}
& P \xrightarrow{(\alpha,r)} P' & (\alpha \notin A) \\
& P \boxtimes_A Q \xrightarrow{(\alpha,r)} P' \boxtimes_A Q & (\alpha \notin A) \\
& P \boxtimes_A Q \xrightarrow{(\alpha,r)} P' \boxtimes_A Q & (\alpha \notin A) \\
& P \boxtimes_Q Q \xrightarrow{(\alpha,R)} Q' & (\alpha \in A) \\
& P \boxtimes A Q \xrightarrow{(\alpha,r_1)} P' \boxtimes A Q' & (\alpha \in A) \\
& P \boxtimes L Q \xrightarrow{(\alpha,r_1)} P' \boxtimes A Q' & (\alpha \in A) \\
& P \boxtimes L Q \xrightarrow{(\alpha,r_2)} Q' & (\alpha \in A) \\
& P \boxtimes A Q \xrightarrow{(\alpha,r_2)} Q' & (\alpha \in A) \\
\end{align*}
\]

where \( R = \frac{r_1}{r_\alpha(P)} \frac{r_2}{r_\alpha(Q)} \min(r_\alpha(P), r_\alpha(Q)) \)
Observation Equivalence

Lumpability on the CTMC

\[ r_{ia} + r_{ib} + r_{id} = r_{jc} + r_{jd} \]

Users cannot distinguish lumpable bisimilar PEPA components
**Observation Equivalence**

**Definition - Lumpable bisimilarity**

It is the largest equivalence relation $\approx_I$ such that if $P \approx_I Q$, then for all $\alpha$ and for each $S$ equivalence class

- either $\alpha \neq \tau$,
- or $\alpha = \tau$ and $P, Q \notin S$,

it holds

$$\sum_{P' \in S, \ P \xrightarrow{(\alpha, r\alpha)} P'} r\alpha = \sum_{Q' \in S, \ Q \xrightarrow{(\alpha, r\alpha)} Q'} r\alpha$$

**Properties**

It is *contextual*, *action preserving*, and induces a *lumpability*. 
**Non-Interference**

**A general definition [Focardi-Gorrieri’95]**

\[ P \in NI \text{ iff } \forall \text{ high level process } H, \ (P | 0) \sim^{low} (P | H) \]

where \( \sim^{low} \) denotes a **low level observation equivalence**

\[ ? \]

Low level user
Stochastic Non-Interference (SNI)

- We partition the actions into $\mathcal{L}$ (low), $\mathcal{H}$ (high), $\{\tau\}$ (sinch.)
- High level processes can only perform high level actions
- Low level users can only perform/observe low level actions

**Definition - SNI**

$$P \in \text{SNI} \iff \forall \text{ high level PEPA component } H$$

$$(P \boxtimes_0 H) \sim^\text{low} (P \boxtimes H)$$

**Low level observation $\sim^\text{low}$**

It is $\approx_1$ without observing actions in $\mathcal{H}$

$$(P \boxtimes_0 H)/\mathcal{H} \approx_1 (P \boxtimes H)/\mathcal{H}$$
**Persistent SNI (PSNI)**

**Definition - PSNI**

\[ P \in \text{PSNI} \text{ iff } \forall \text{ derivative } P' \text{ of } P \]

\[ P' \in \text{SNI} \]
Toy Example: Unsecure Vs Secure System

Unsecure

Secure iff $\alpha_4 = \alpha_5$
Delimited PSNI (D_PSNI)

- We partition the actions into $\mathcal{L}$, $\mathcal{H}$, $\mathcal{D}$ (downgrading), $\{\tau\}$
- Downgrading actions specify the behavior of a trusted component that allows delimited flows from high to low
- Low level users can only perform/observe low level actions

Definition - D_PSNI

$P \in D_{PSNI}$ iff $\forall$ derivative $P'$ of $P$

$\forall$ high level PEPA component $H$

$((P' \boxdot H)/\mathcal{H}) \\downarrow \mathcal{D} \approx (P'/\mathcal{H})/\mathcal{H} \\downarrow \mathcal{D}$
THE IMPORTANCE OF PERSISTENCE

EXAMPLE

\[ P \rightarrow P' \rightarrow P'' \rightarrow 0 \]

\( P \) satisfies the condition, while \( P' \) does not

INTUITIVELY

\[ \text{The } d \text{ action downgrades the high incoming actions} \]

\[ \text{It does not downgrade subsequent high actions} \]
Let us . . .

. . . focus on $PSNI$

Luckily, as for the secure process algebra, $D_{PSNI}$ is mainly a technical generalization
Properties

Theorem - Unwinding

\[ P \in \text{PSNI} \text{ iff } \forall \text{ derivative } P' \text{ of } P, \]

\[ P' \xrightarrow{(h,r)} P'' \text{ implies } P' \setminus \mathcal{H} \approx \approx_1 P'' \setminus \mathcal{H} \]
Properties

Theorem - Unwinding

\[ P \in PSNI \text{ iff } \forall \text{ derivative } P' \text{ of } P, \]

\[ P' \xrightarrow{(h,r)} P'' \text{ implies } P' \backslash \mathcal{H} \approx_1 P'' \backslash \mathcal{H} \]

- This allows to explicitly identify the dangerous situations.
- Whenever a high level action is performed we impose syntactic conditions that ensure \( \approx_1 \).
Properties

Theorem - Compositionality I

Let $P, P_i \in PSNI$, $Q$ be a PEPA component, and $A \subseteq \mathcal{A} \setminus \{\tau\}$
The following processes are $PSNI$

- $0$
- $Q \setminus \mathcal{H}$, $Q \setminus \mathcal{L}$, $Q/\mathcal{H}$, and $Q/\mathcal{L}$
- $(\ell, r).P$ with $\ell \in \mathcal{L} \cup \{\tau\}$
- $P/A$ and $P \setminus A$
- $P_i \downarrow_{A} P_j$
Theorem - Compositionality I

Let \( P, P_i \in \text{PSNI}, \ Q \) be a PEPA component, and \( A \subseteq \mathcal{A} \setminus \{\tau\} \)

The following processes are \( \text{PSNI} \):

1. \( 0 \)
2. \( Q \setminus \mathcal{H}, Q \setminus \mathcal{L}, Q / \mathcal{H}, \) and \( Q / \mathcal{L} \)
3. \( (\ell, r).P \) with \( \ell \in \mathcal{L} \cup \{\tau\} \)
4. \( P / A \) and \( P \setminus A \)
5. \( P_i \blacktriangleright A P_j \)

Remark

These are consequences of \( \text{PEPA broadcasting synchronization rules} \) and are not true in other process algebra (e.g., CCS like)
Properties

Theorem - Compositionality II

Let \( P, P_i \in PSNI \), \( Q \) be a PEPA component, and \( A \subseteq A \setminus \{ \tau \} \)

- \( X_c, X'_c \) are \( PSNI \) where

\[
X_c \overset{\text{def}}{=} \sum_{i \in I} (\ell_i, r_i).P_i + \sum_{k \in K} (\ell_k, r_k).X_k + \sum_{j \in J} (h_j, r_j).X_c \setminus H_j + \sum_{m \in M} (h_m, r_m).X'_c
\]

\[
X'_c \overset{\text{def}}{=} \sum_{i \in I} (\ell_i, r_i).P_i + \sum_{k \in K} (\ell_k, r_k).X_k
\]

Remark

\( \text{▶ } \) This is a trade-off between readability and expressivity

\( \text{▶ } \) How much can we improve? See Some of My Favourite Results in Classic Process Algebra by L. Aceto
**THEOREM - COMPOSITIONALITY II**

Let \( P, P_i \in PSNI, \ Q \) be a PEPA component, and \( A \subseteq A \setminus \{\tau\} \)

- \( X_c, X'_c \) are \( PSNI \) where

\[
X_c \overset{\text{def}}{=} \sum_{i \in I} (\ell_i, r_i).P_i + \sum_{k \in K} (\ell_k, r_k).X_k + \sum_{j \in J} (h_j, r_j).X_c \setminus H_j + \sum_{m \in M} (h_m, r_m).X'_c
\]

\[
X'_c \overset{\text{def}}{=} \sum_{i \in I} (\ell_i, r_i).P_i + \sum_{k \in K} (\ell_k, r_k).X_k
\]

**REMARK**

- This is a trade-off between readability and expressivity
- How much can we improve? See *Some of My Favourite Results in Classic Process Algebra* by L. Aceto
**PSNI Process Algebra**

**Definition - \( C_{PSNI} \)**

Let \( Q \) be PEPA component and \( A \subseteq A \setminus \{\tau\} \)
\( C_{PSNI} \) is defined by the following grammar:

\[
S ::= 0 \mid Q \setminus H \mid Q \setminus L \mid (\ell, r).S \mid X
\]

\[
P ::= S \mid P/A \mid P \setminus A \mid P \oslash_A P
\]

where \( X \) has a recursive definition of the form

\[
X \overset{\text{def}}{=} \sum_{i \in I} (\ell_i, r_i).S_i + \sum_{j \in J} (h_j, r_j).X \setminus H_j + \sum_{m \in M} (h_m, r_m).X'
\]

\[
X' \overset{\text{def}}{=} \sum_{i \in I} (\ell_i, r_i).S_i
\]
**PSNI Process Algebra**

**Definition - \( C_{PSNI} \)**

Let \( Q \) be PEPA component and \( A \subseteq A \setminus \{\tau\} \)

\( C_{PSNI} \) is defined by the following grammar:

\[
S ::= 0 \mid Q \setminus H \mid Q \setminus L \mid (\ell, r).S \mid X
\]

\[
P ::= S \mid P/A \mid P \setminus A \mid P \boxtimes_A P
\]

where \( X \) has a recursive definition of the form

\[
X \overset{\text{def}}{=} \sum_{i \in I} (\ell_i, r_i).S_i + \sum_{j \in J} (h_j, r_j).X \setminus H_j + \sum_{m \in M} (h_m, r_m).X'
\]

\[
X' \overset{\text{def}}{=} \sum_{i \in I} (\ell_i, r_i).S_i
\]

**Remark**

- We can also define infinite state processes
- We can generalize to a process algebra for \( D_{PSNI} \)
Toy Examples: $C_{PSNI}$ Systems
CONCLUSION

- A general framework for PSNI and D_PSNI has been presented.
- The use of Contextual Lumpability guarantees that the steady state distribution is not influenced by the high level behavior.
- Two process algebras that allow to define processes secure by construction have been introduced.

QUESTIONS

- Can we find a complete process algebra?
- How is it related to efficient computation of lumpability/bisimulation?