Synthesis of non-interferent systems

Gilles Benattar[†] Franck Cassez[‡] Didier Lime[†]
Olivier H.Roux[†]

[†]IRCCyN/CNRS UMR 6597, Nantes, France

[‡]CNRS and National ICT Australia, Sydney, Australia

Formal Modelling and Analysis of Timed Systems 2009
(FORMATS09)

Introduction

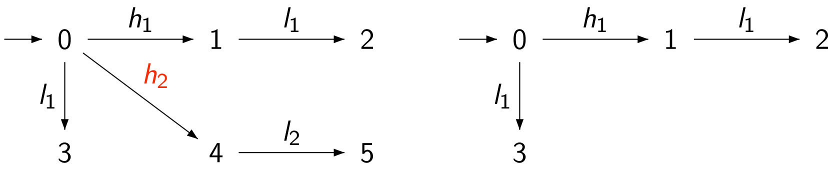
1. Studies of information flow security properties has been a very active domain.
2. *Information flow analysis* defines secrecy as: “high level information never flows into low level channels”
   i.e., *non-interference*.
3. There are many results on model checking of non-interference properties.
4. We consider the problem of the *synthesis* of non-interferent systems for timed and untimed automata.

Restriction definition

$$\begin{tikzcd} {} \arrow[r] & 0 \arrow[r, "h_1"] \arrow[d, "l_1"'] \arrow[rd, "h_2"] & 1 \arrow[r, "l_1"] & 2 \\ & 3 & 4 \arrow[r, "l_2"] & 5 \end{tikzcd}$$

Figure: $\mathcal{B}$

Restriction definition



(a) Automaton $\mathcal{B}$

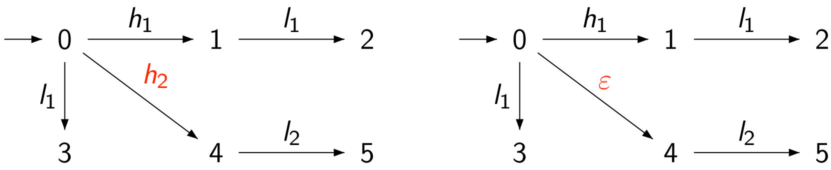
(b) $\mathcal{B}\backslash\{h_2\}$

Abstraction (hiding) definition

$$\begin{tikzcd}
{} \arrow[r] & 0 \arrow[r, "h_1"] \arrow[d, "l_1"'] \arrow[dr, "h_2"] & 1 \arrow[r, "l_1"] & 2 \\
& 3 & 4 \arrow[r, "l_2"] & 5
\end{tikzcd}$$

Figure: $\mathcal{B}$

Abstraction (hiding) definition



(a) Automaton $\mathcal{B}$

(b) $\mathcal{B}/\{h_2\}$

Strong Non-deterministic Non-Interference (SNNI) 1/4

1. The systems is defined by an automaton $\mathcal{A}$ over an alphabet Σ divided into two sub-alphabets : Σ_h the *high level actions* and Σ_l the *low level actions*
2. A system defined by an automaton $\mathcal{A}$ is *non-interferent* if the low level user cannot distinguish $\mathcal{A}/\Sigma_h$ from $\mathcal{A}\backslash\Sigma_h$.

Definition (SNNI)

A TA $\mathcal{A}$ has the *strong non-deterministic non-interference* property (in short "$\mathcal{A}$ is SNNI") if $\mathcal{A}/\Sigma_h \approx_{\mathcal{L}} \mathcal{A}\backslash\Sigma_h$, where $A_1 \approx_{\mathcal{L}} A_2$ mean that A_1 and A_2 are *language equivalent*.

SNNI finite automata example 1/2

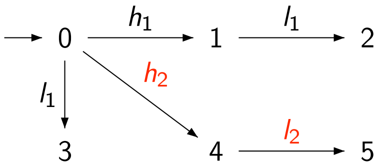


Figure: $\mathcal{B}$ that is not SNNI

- $\mathcal{L}(\mathcal{B}/\{h_1, h_2\}) = \{l_1, l_2\}$
- $\mathcal{L}(\mathcal{B}\backslash\{h_1, h_2\}) = \{l_1\}$

SNNI finite automata example 2/2

$$\begin{tikzcd}
{} \arrow[r] & 0 \arrow[r, "h_1"] \arrow[d, "l_1"'] \arrow[dr, "h_2"] & 1 \arrow[r, "l_1"] & 2 \\
& 3 & 4 &
\end{tikzcd}$$

Figure: $\mathcal{C}$ that is SNNI

- $\mathcal{L}(\mathcal{C}/\{h_1, h_2\}) = \{l_1\}$
- $\mathcal{L}(\mathcal{C}\backslash\{h_1, h_2\}) = \{l_1\}$

SNNI timed automata example

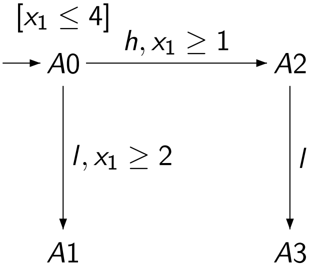


Figure: Timed Automaton $\mathcal{A}$

SNNI timed automata example

$$\begin{tikzcd}
\arrow[r] & A0 \arrow[rr, "h"] \arrow[d, "l"] & & A2 \arrow[d, "l"] \\
& A1 & & A3
\end{tikzcd}$$

Figure: Finite Automaton $\mathcal{A}' = \mathit{untimed}(\mathcal{A})$

- $\mathcal{L}(\mathcal{A}'/\{h\}) = \{l\}$
- $\mathcal{L}(\mathcal{A}' \backslash \{h\}) = \{l\}$

SNNI timed automata example

$$\begin{tikzcd}
\arrow[r] & A0 \arrow[r, "{h, x_1 \geq 1}"] \arrow[d, "{l, x_1 \geq 2}"] & A2 \arrow[d, "l"] \\
& A1 & A3
\end{tikzcd}$$

$[x_1 \leq 4]$

Figure: Timed Automaton $\mathcal{A}$

$$\rho = (A0, 0) \xrightarrow{1.1} (A0, 1.1) \xrightarrow{h} (A2, 0) \xrightarrow{0.5} (A2, 1.6) \xrightarrow{l} (A3, 1.6) \in Runs(\mathcal{A})$$

SNNI timed automata example

$$
\begin{tikzcd}
\arrow[r] & \underset{}{\overset{[x_1 \leq 4]}{A0}} \arrow[rr, "{h, x_1 \geq 1}"] \arrow[d, "{l, x_1 \geq 2}"] & & A2 \arrow[d, "l"] \\
& A1 & & A3
\end{tikzcd}
$$

Figure: Timed Automaton $\mathcal{A}$

$$\rho = (A0, 0) \xrightarrow{1.1} (A0, 1.1) \xrightarrow{h} (A2, 0) \xrightarrow{0.5} (A2, 1.6) \xrightarrow{l} (A3, 1.6) \in Runs(\mathcal{A})$$

$$(1.1, h).(0.5, l) \in \mathcal{L}(\mathcal{A})$$

SNNI timed automata example

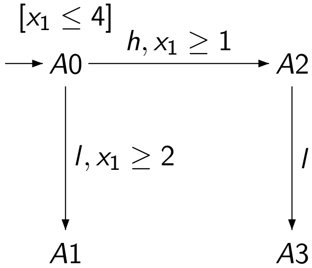


Figure: Timed Automaton $\mathcal{A}$

$$\rho = (A0, 0) \xrightarrow{1.1} (A0, 1.1) \xrightarrow{h} (A2, 0) \color{red}{\xrightarrow{0.5} (A2, 1.6)} \color{red}{\xrightarrow{l}}\color{black}{(A3, 1.6) \in Runs(\mathcal{A})}$$

$$(1.1, h).\color{red}{(0.5, l)}\color{black}{} \in \mathcal{L}(\mathcal{A}) \Rightarrow \color{red}{(1.6, l)}\color{black}{} \in \mathcal{L}(\mathcal{A}/\{h\})$$

SNNI timed automata example

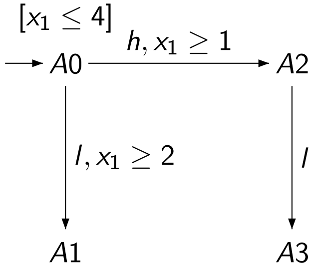


Figure: Timed Automaton $\mathcal{A}$

$$\rho = (A0, 0) \xrightarrow{1.1} (A0, 1.1) \xrightarrow{h} (A2, 0) \xrightarrow{0.5} (A2, 1.6) \xrightarrow{l} (A3, 1.6) \in Runs(\mathcal{A})$$

$$(1.1, h).(0.5, l) \in \mathcal{L}(\mathcal{A}) \Rightarrow (1.6, l) \in \mathcal{L}(\mathcal{A}/\{h\}) \Rightarrow \mathcal{A} \text{ is not SNNI}$$

Control problem 1/2

- The *SNNI Verification Problem* (SNNI-VP) for a system S asks the following: is S SNNI ?
- The *Control Problem* (SNNI-CP) for a system S asks the following: Is there a controller C s.t. $C(S)$ is SNNI ?
- The *Controller Synthesis Problem* (SNNI-CSP) asks to compute a witness controller C.

Control problem 2/2

Let $\Sigma_c \subseteq \Sigma = \Sigma_h \cup \Sigma_l$ a set of *controllable actions*, let $\lambda \notin \Sigma$ the *waiting action*.

Definition (Controller)

A *controller* C for $\mathcal{A}$ is a partial mapping $C : Runs(\mathcal{A}) \to 2^{\Sigma_c \cup \{\lambda\}}$.

- After each run $\rho \in Runs(\mathcal{A})$, the controller chose a set $C(\rho)$ of actions that are not disabled.

Control problem 2/2

Let $\Sigma_c \subseteq \Sigma = \Sigma_h \cup \Sigma_l$ a set of *controllable actions*, let $\lambda \notin \Sigma$ the *waiting action*.

Definition (Controller)

A *controller* C for $\mathcal{A}$ is a partial mapping $C : Runs(\mathcal{A}) \to 2^{\Sigma_c \cup \{\lambda\}}$.

- After each run $\rho \in Runs(\mathcal{A})$, the controller chose a set $C(\rho)$ of actions that are not disabled.
- If $\lambda \in C(\rho)$, the system may wait, otherwise, a controllable action must be done by one of the users.

SNNI Verification Problem (SNNI-VP)

	Untimed Automata	Timed Automata
Deterministic $A\backslash\Sigma_h$	PTIME	PSPACE-Complete
Non-deterministic $A\backslash\Sigma_h$	PSPACE-Complete	Undecidable [1]

Table: Results for the SNNI-VP

SNNI Control Problem (SNNI-CP) for finite automata 1/2

Theorem

For finite automata, the SNNI-CP is PSPACE-Complete.

SNNI Control Problem (SNNI-CP) for finite automata 2/2

For finite automata, we can easily check if SNNI is controllable by cutting all the controllable actions and checking if the obtained system is SNNI.

$$\begin{tikzcd} & {} \arrow[r] & 0 \arrow[r, "h_1"] \arrow[d, "l_1"] & 4 \arrow[r, "l_1"] & 5 \\ 3 & 2 \arrow[l, "l_2"'] & 1 \arrow[l, "h_2"] & & \end{tikzcd}$$

Figure: Automaton $\mathcal{D}$

- $\Sigma_c = \{l_1\}$

SNNI Control Problem (SNNI-CP) for finite automata 2/2

For finite automata, we can easily check if SNNI is controllable by cutting all the controllable actions and checking if the obtained system is SNNI.

$$\longrightarrow 0 \xrightarrow{h_1} 4$$

Figure: Automaton $\mathcal{D} \backslash \Sigma_c$

- $\Sigma_c = \{l_1\}$

SNNI Control Problem (SNNI-CP) for timed automata

This does not work in the timed case :

$$
\begin{tikzcd}
\arrow[r] & 0 \arrow[r, "{h, x_1 \geq 5}"] \arrow[d, "{a, x_1 > 1}"] & 2 \arrow[d, "b"] \\
& 1 & 3
\end{tikzcd}
$$

Figure: Timed Automaton $\mathcal{E}$

- $\Sigma_c = \{a\}$

SNNI Control Problem (SNNI-CP) for timed automata

This does not work in the timed case :

$$\begin{tikzcd}
{} \arrow[r] & 0 \arrow[r, "{h, x_1 \geq 5}"] \arrow[d, "{a, x_1 > 1}"] & 2 \arrow[d, "b"] \\
& 1 & 3
\end{tikzcd}$$

(a) Timed Automaton $\mathcal{E}$

$$\begin{tikzcd}
{} \arrow[r] & 0 \arrow[r, "{h, x_1 \geq 5}"] & 2 \arrow[d, "b"] \\
& & 3
\end{tikzcd}$$

(b) Timed Automaton $\mathcal{E} \backslash \Sigma_c$

- $\Sigma_c = \{a\}$

SNNI Control Problem (SNNI-CP) for timed automata

This does not work in the timed case :

$$\begin{tikzcd} {} \arrow[r] & 0 \arrow[r, "h, x_1 \geq 5"] \arrow[d, "a, x_1 > 1"] & 2 \arrow[d, "b"] \\ & 1 & 3 \end{tikzcd}$$

(c) Automaton $\mathcal{E}$

$$\begin{tikzcd} & {[x_1 \leq 4]} \\ {} \arrow[r] & 0 \arrow[d, "a, x_1 > 1"] \\ & 1 \end{tikzcd}$$

(d) Timed Automaton $C(\mathcal{E})$

- $\Sigma_c = \{a\}$

SNNI Controller Synthesis Problem (SNNI-CSP)

Theorem

If $\mathcal{A}$ is a finite automaton, we can compute the most permissive controller C s.t. $C(\mathcal{A})$ is SNNI.

Theorem

If $\mathcal{A}$ is a timed automaton and $\mathcal{A}\backslash\Sigma_h$ is deterministic, we can compute the most permissive controller C s.t. $C(\mathcal{A})$ is SNNI.

SNNI Controller Synthesis Problem (SNNI-CSP) for finite automata 1/5

Let $\mathcal{D}$ be an automaton. In order to solve the SNNI-CSP, we calculate iteratively the most permissive controller of safety games calculated from $\mathcal{D}$ and $\mathcal{D}\backslash\Sigma_h$.

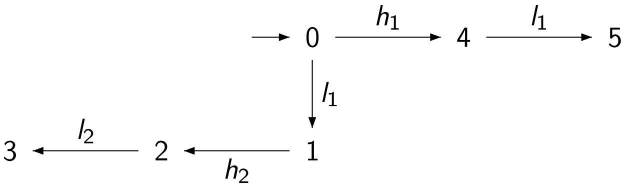


Figure: Timed Automaton $\mathcal{D} = \mathcal{D}^0$

- $\Sigma_c = \{l_1, h_1\}$

SNNI Controller Synthesis Problem (SNNI-CSP) for finite automata 2/5

We define $\mathcal{D}_2$ as the *complete* version of $\mathcal{D} \backslash \Sigma_h$.

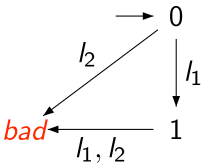


Figure: Automaton $\mathcal{D}_2^0$

SNNI Controller Synthesis Problem (SNNI-CSP) for finite automata 3/5

We compute $\mathcal{D}^0 \otimes \mathcal{D}_2^0$, and define a controller $C_1^{\otimes}$ that solves the safety game.

$$\begin{tikzcd}
 & & \arrow[r] & 00 \arrow[r, "h_1"] \arrow[d, "l_1"] & 40 \arrow[r, "l_1"] & 51 \\
\textcolor{red}{3bad} & 21 \arrow[l, "l_2"'] & & 11 \arrow[ll, "h_2"] & &
\end{tikzcd}$$

Figure: Automaton $\mathcal{D}^0 \otimes \mathcal{D}_2^0$

- $\Sigma_c = \{l_1, h_1\}$

SNNI Controller Synthesis Problem (SNNI-CSP) for finite automata 3/5

We compute $\mathcal{D}^0 \otimes \mathcal{D}_2^0$, and define a controller $C_1^{\otimes}$ that solves the safety game.

$$
\begin{tikzcd}
 & & {} \arrow[r] & 00 \arrow[r, "h_1"] \arrow[d, "l_1"] & 40 \arrow[r, "l_1"] & 51 \\
3bad & 21 \arrow[l, "l_2"'] & & 11 \arrow[ll, "h_2"'] & &
\end{tikzcd}
$$

Figure: Timed Automaton $\mathcal{D}_p^0 = \mathcal{D}^0 \otimes \mathcal{D}_2^0$

- $\Sigma_c = \{l_1, h_1\}$

SNNI Controller Synthesis Problem (SNNI-CSP) for finite automata 3/5

We compute $\mathcal{D}^0 \otimes \mathcal{D}_2^0$, and define a controller $C_1^{\otimes}$ that solves the safety game.

$$\longrightarrow 00 \xrightarrow{h_1} 40 \xrightarrow{l_1} 51$$

Figure: Timed Automaton $C_1^{\otimes}(\mathcal{D} \otimes \mathcal{D}_2)$

- $\Sigma_c = \{l_1, h_1\}$

SNNI Controller Synthesis Problem (SNNI-CSP) for finite automata 4/5

We compute C^1 from $C_1^{\otimes}$ and if $\mathcal{L}(C^1(\mathcal{D}^0)\backslash\Sigma_h) \neq \mathcal{L}(\mathcal{D}^0)\backslash\Sigma_h)$, we iterate process.

$$\longrightarrow 0 \xrightarrow{h_1} 4 \xrightarrow{l_1} 5$$

Figure: Timed Automaton $C^1(\mathcal{D})$

SNNI Controller Synthesis Problem (SNNI-CSP) for finite automata 5/5

We reach a fix point C^*

$$\longrightarrow 0 \xrightarrow{\quad h_1 \quad} 1$$

Figure: Timed Automaton $C^*(\mathcal{D})$ that is SNNI

SNNI Controller Synthesis Problem (SNNI-CSP) for timed automata

We proved that the same algorithm works for a timed automaton $\mathcal{A}$ if $\mathcal{A}\backslash\Sigma_h$ is deterministic.

$$\begin{tikzcd}
 & \overset{[x_1 \leq 4]}{A0} \arrow[r, "{h, x_1 \geq 1}"] \arrow[d, "{l, x_1 \geq 2}"] & A2 \arrow[d, "l"] \\
 \arrow[ur, phantom] & A1 & A3
\end{tikzcd}$$

Figure: Timed Automaton $\mathcal{A}$

- $\Sigma_c = \{l\}$

SNNI Controller Synthesis Problem (SNNI-CSP) for timed automata

We proved that the same algorithm works for a timed automaton $\mathcal{A}$ if $\mathcal{A}\backslash\Sigma_h$ is deterministic.

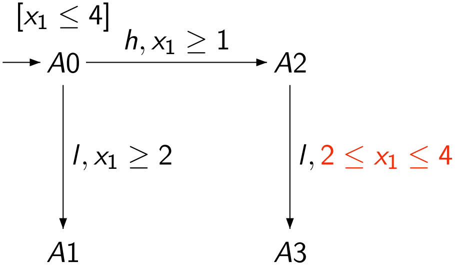


Figure: Timed Automaton $C^*(\mathcal{A})$

1. Introduction
2. Definitions
   - Preliminaries
   - Non-interference
   - Control problem
3. Results
   - SNNI verification problem
   - SNNI control problem
   - SNNI control synthesis problem
4. Conclusion

Conclusion

	A Timed Automaton		A Finite Automaton	
	$A \backslash \Sigma_h$ Non-Det.	$A \backslash \Sigma_h$ Det.	$A \backslash \Sigma_h$ Non-Det.	$A \backslash \Sigma_h$ Det.
SNNI-VP	undecidable [1]	PSPACE-C	PSPACE-C	PTIME
SNNI-CP	undecidable [1]	EXPTIME-C	PSPACE-C	PTIME
SNNI-CSP	undecidable [1]	EXPTIME-C	EXPTIME [2]	PSPACE-C

Table: Summary of the Results

Future works

1. Extend the results on other form of non-interference (CSNNI and BSNNI) for untimed and timed automata.
2. Determine conditions under which a most permissive controller exists for the BSNNI-CSP and CSNNI-CSP

Thanks

Thank you for your attention

Bibliography

Gardey, G., Mullins, J., Roux, O.H.:
Non-interference control synthesis for security timed automata.

Elec. Notes in Theo. Comp. Science **180**(1) (2005) 35–53. Proceedings of the 3rd International Workshop on Security Issues in Concurrency (SecCo'05).

Cassez, F., Mullins, J., Roux, O.H.:
Synthesis of non-interferent systems.
In: Proceedings of the 4th Int. Conf. on Mathematical Methods, Models and Architectures for Computer Network Security (MMM-ACNS'07). Volume 1 of Communications in Computer and Inform. Science, Springer (2007) 307–321.