Partial Order Reduction for the applied $\pi$-calculus

SEQUOIA

Lucca Hirschi

LSV, ENS Cachan

March 9, 2015

joint work with

David Baelde and Stéphanie Delaune

LSV and LSV
unsecure network + active attacker → (tricky) attacks

we need formal verification of crypto protocols
unsecure network + active attacker \rightarrow (tricky) attacks

\implies we need formal \textit{verification} of crypto protocols

\textbf{Our setting}

- Applied-$\pi$ models protocols;
- Trace equivalence models security properties.
Introduction

unsecure network + active attacker $\rightarrow$ (tricky) attacks

$\xrightarrow{\sim}$ we need formal verification of crypto protocols

Our setting

- Applied-$\pi$ models protocols;
- Trace equivalence models security properties.

$\xrightarrow{\sim}$ several algorithms resolve this problem (Akiss, Apte, Spec)

Issue: Limited practical impact

Too slow. Bottleneck: size of search space (interleavings).
Outline

1. Introduction
2. Model
3. Big Picture
4. Compression
5. Reduction
6. Conclusion
Outline

1 Introduction
2 Model
3 Big Picture
4 Compression
5 Reduction
6 Conclusion
Applied-$\pi$ - Syntax

Terms

$\mathcal{T}$: set of terms + equational theory. e.g., $\text{dec}({m}_k, k) =_E m$. 
## Applied-$\pi$ - Syntax

### Terms

$\mathcal{T}$: set of terms + equational theory. e.g., $\text{dec}\{m\}_k, k \equiv_E m$.

### Processes and configurations

\[
P, Q ::= 0 \mid (P \mid Q) \mid \text{in}(c, x).P \mid \text{out}(c, m).P \mid \text{if } u = v \text{ then } P \text{ else } Q \mid !a_{c, \vec{n}} P
\]

think of $!\nu \vec{c}.\nu \vec{n}.\text{out}(a, \vec{c}).P$
Applied-$\pi$ - Syntax

Terms

$\mathcal{T}$: set of terms + equational theory. e.g., $\text{dec}(\{m\}_{k}, k) \equiv_{E} m$.

Processes and configurations

$P, Q ::= 0 \mid (P\mathbin{|}Q) \mid \text{in}(c, x).P \mid \text{out}(c, m).P \mid \text{if } u = v \text{ then } P \text{ else } Q \mid \text{!}_a \overset{c}{\to} \overset{n}{\to} P$

think of $\text{!}_v \overset{c}{\to} \overset{n}{\to} \text{out}(a, \overset{c}{\to}) . P$

Internal reduction $\rightsquigarrow$

- $(\text{if } u = v \text{ then } P \text{ else } Q) \rightsquigarrow P$ when $u \equiv_{E} v$;
- $(\text{if } u = v \text{ then } P \text{ else } Q) \rightsquigarrow Q$ when $u \not\equiv_{E} v$;
- $(P \mathbin{|} Q) \rightsquigarrow (P' \mathbin{|} Q)$ and $(Q \mathbin{|} P) \rightsquigarrow (Q \mathbin{|} P')$ when $P \rightsquigarrow P'$;
- $((P_1 \mathbin{|} P_2) \mathbin{|} P_3) \rightsquigarrow (P_1 \mathbin{|} (P_2 \mathbin{|} P_3))$; notation $\Pi_{i=1}^{3} P_i$
- $(P \mathbin{|} 0) \rightsquigarrow P$ and $(0 \mathbin{|} P) \rightsquigarrow P$. 
Applied-\(\pi\) - Semantics

\[ \text{IN} \quad (\{\text{in}(c, x).Q\} \uplus P; \Phi) \xrightarrow{\text{in}(c,M)} (\{Q\{u/x\}\} \uplus P; \Phi) \]
where \(M \in T(\text{dom}(\Phi))\) and \(M\Phi =_E u\)

\[ \text{OUT} \quad (\{\text{out}(c, u).Q\} \uplus P; \Phi) \xrightarrow{\text{out}(c,w)} (\{Q\} \uplus P; \Phi \cup \{w \mapsto u\}) \]
where \(w \in \mathcal{W}\) is fresh

\[ \text{PAR} \quad (\{\prod_{i=1}^n P_i\} \uplus P; \Phi) \xrightarrow{\text{par}} (\{P_1; \ldots; P_n\} \uplus P; \Phi) \]

\[ \text{ZERO} \quad (\{0\} \uplus P; \Phi) \xrightarrow{\text{zero}} (P; \Phi) \]
Applied-\(\pi\) - Semantics

\[
\text{IN} \quad (\{\text{in}(c, x).Q\} \uplus P; \Phi) \xrightarrow{\text{in}(c,M)} (\{Q\{u/x\}\} \uplus P; \Phi)
\]
where \(M \in \mathcal{T}(\text{dom}(\Phi))\) and \(M\Phi =_E u\)

\[
\text{OUT} \quad (\{\text{out}(c, u).Q\} \uplus P; \Phi) \xrightarrow{\text{out}(c,w)} (\{Q\} \uplus P; \Phi \cup \{w \mapsto u\})
\]
where \(w \in \mathcal{W}\) is fresh

\[
\text{PAR} \quad (\prod_{i=1}^{n} P_i) \uplus P; \Phi \xrightarrow{\text{par}} (\{P_1; \ldots; P_n\} \uplus P; \Phi)
\]

\[
\text{ZERO} \quad (\{0\} \uplus P; \Phi) \xrightarrow{\text{zero}} (P; \Phi)
\]

\[
\text{REPL} \quad (\{!a_{\overrightarrow{c}}, \overrightarrow{n}.P\} \uplus P; \Phi) \xrightarrow{\text{sess}(a, \overrightarrow{c})} (P; !a_{\overrightarrow{c}}, \overrightarrow{n}.P) \uplus P; \Phi)
\]
where \(\overrightarrow{c}, \overrightarrow{n}\) are fresh
Applied-$\pi$ - Semantics

<table>
<thead>
<tr>
<th>Rule</th>
<th>Expression</th>
<th>Transition</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>${\text{in}(c, x).Q} \cup P; \Phi$</td>
<td>$\text{in}(c, M)$</td>
<td>where $M \in T(\text{dom}(\Phi))$ and $M\Phi = E u$</td>
</tr>
<tr>
<td>OUT</td>
<td>${\text{out}(c, u).Q} \cup P; \Phi$</td>
<td>$\text{out}(c, w)$</td>
<td>where $w \in \mathcal{W}$ is fresh</td>
</tr>
<tr>
<td>PAR</td>
<td>${\Pi_{i=1}^{n} P_i} \cup P; \Phi$</td>
<td>$\text{par}$</td>
<td>${P_1; \ldots; P_n} \cup P; \Phi$</td>
</tr>
<tr>
<td>ZERO</td>
<td>${0} \cup P; \Phi$</td>
<td>$\text{zero}$</td>
<td>${P; \Phi}$</td>
</tr>
<tr>
<td>REPL</td>
<td>${a\overrightarrow{c}, \overrightarrow{n}.P} \cup P; \Phi$</td>
<td>$\text{sess}(a, \overrightarrow{c})$</td>
<td>where $\overrightarrow{c}, \overrightarrow{n}$ are fresh</td>
</tr>
</tbody>
</table>

Trace equivalence

$A \approx B \iff \forall A \xrightarrow{t} A', \exists B \xrightarrow{t'} B' \text{ such that } \text{obs}(t) = \text{obs}(t') \text{ and } \Phi_{A'} \sim \Phi_{B'}$ (and conversely)

$\Phi \sim \Phi' \iff (\forall M, N, M\Phi = N\Phi \iff M\Phi' = N\Phi')$
Big Picture

- **Motivation:** Improve algorithms checking trace equivalence
- **How:** Remove redundant interleavings via a reduced semantics

\[ \rightarrow \approx \rightarrow \approx_c \rightarrow \approx_r \]

Theorem 1: \( \approx = \approx_c \)

Theorem 2: \( \approx_c = \approx_r \)

\( \rightarrow_r \) does not explore all behaviours but sufficiently to ensure \( \approx = \approx_r \)
Big Picture

\[ \rightarrow \approx \quad \text{Compression} \quad \Rightarrow \quad \text{Theorem 1: } \approx = \approx_c \]

\[ \rightarrow c \approx_c \quad \text{Reduction} \quad \Rightarrow \quad \text{Theorem 2: } \approx_c = \approx_r \]

\[ \rightarrow r \approx_r \]

Required properties

\( \rightarrow_r \) is such that:

- reachability properties coincide on \( \rightarrow_r \) and \( \rightarrow \);
- for action-determinate processes, trace-equivalence coincides on \( \rightarrow_r \) and \( \rightarrow \).
Big Picture

Theorem 1:
\[ \approx = \approx_c \]
Reduction

Theorem 2:
\[ \approx_c = \approx_r \]

Required properties

\( \rightarrow_r \) is such that:

- reachability properties coincide on \( \rightarrow_r \) and \( \rightarrow \);
- for action-determinate processes, trace-equivalence coincides on \( \rightarrow_r \) and \( \rightarrow \).

Action-determinism

\( A \) is action-deterministic if \( \forall A \xrightarrow{t} (P; \Phi), \forall P, Q \in P \), \( P \) and \( Q \) cannot perform an observable action of the same nature on the same channel.

Attacker knows to/from whom he is sending/receiving messages.
The Idea

Follow a particular strategy that reduces the number of choices by looking at the nature of available actions.

Polarities of processes:

- **negative**: `\text{out}().P, \Pi P_i, 0`
  - Bring new data or choices, execution independent on the context
The Idea

Follow a particular **strategy** that reduces the number of choices by looking at the **nature** of available actions.

**Polarities of processes:**

- **negative:** `out().P, ΠP_i, 0`
  Bring new data or choices, execution **independent** on the context

- **positive:** `in().P`
  Execution **depends** on the context
Intuitions

The Idea

Follow a particular **strategy** that reduces the number of choices by looking at the **nature** of available actions.

Polarities of processes:

- **negative**: $\text{out}(\cdot).P, \Pi P_i, 0$
  - Bring new data or choices, execution **independent** on the context
  - $\leadsto$ to be performed as soon as possible in a given order

- **positive**: $\text{in}(\cdot).P$
  - Execution **depends** on the context
Intuitions

The Idea

Follow a particular **strategy** that reduces the number of choices by looking at the **nature** of available actions.

**Polarities of processes:**

- **negative:** `out().P, ΠP_i, 0`
  Bring new data or choices, execution **independent** on the context
  \(\rightsquigarrow\) to be performed as soon as possible in a given order

- **positive:** `in().P`
  Execution **depends** on the context
  \(\rightsquigarrow\) can be performed only if no **negative**
Intuitions

The Idea

Follow a particular **strategy** that reduces the number of choices by looking at the **nature** of available actions.

Polarities of processes:

- **negative**: `out().P, \Pi P_i, 0`
  - Bring new data or choices, execution **independent** on the context
  - \(\Rightarrow\) to be performed as soon as possible in a given order

- **positive**: `in().P`
  - Execution **depends** on the context
  - \(\Rightarrow\) can be performed only if no **negative**
  - \(\Rightarrow\) we make a choice that we must maintain while it is **positive**
  - \(\Rightarrow\) the chosen one is **under focus**, released when **negative**
Intuitions

The Idea
Follow a particular **strategy** that reduces the number of choices by looking at the **nature** of available actions.

Polarities of processes:

- **negative**: `\textbf{out}().P, \Pi P_i, 0`
  Bring new data or choices, execution **independent** on the context
  \xrightarrow{} to be performed as soon as possible in a given order

- **positive**: `\textbf{in}().P`
  Execution **depends** on the context
  \xrightarrow{} can be performed only if no **negative**
  \xrightarrow{} we make a choice that we must maintain while it is **positive**
  \xrightarrow{} the chosen one is **under focus**, released when **negative**

(Replication: `!a \xrightarrow{c, \vec{n}} P` is **negative** but can start + phase)
Intuitions

The Idea

Follow a particular **strategy** that reduces the number of choices by looking at the **nature** of available actions.

**Polarities of processes:**

- **negative**: $\text{out()}.P, \Pi P_i, 0$
  - Bring new data or choices, execution **independent** on the context
  - $\Rightarrow$ to be performed as soon as possible in a given order

- **positive**: $\text{in()}.P$
  - Execution **depends** on the context
  - $\Rightarrow$ can be performed only if no **negative**
  - $\Rightarrow$ we make a choice that we must maintain while it is **positive**
  - $\Rightarrow$ the chosen one is **under focus**, released when **negative**

(Replication: $!_a^{\tilde{c}} \longrightarrow P$ is **negative** but can start $+$ phase)

If **positive** phase ends with a 0: stop the execution.

Ex: $\text{in}(ca, X).0 \mid \text{in}(cb, Y).0 \mid Q$

$$P \xrightarrow{\text{in}(ca, X), \text{in}(cb, Y)}_{ca}$$
Compressed semantics - Example

Wide Mouthed frog

<table>
<thead>
<tr>
<th>Alice → Server</th>
<th>Server → Bob</th>
<th>Bob →</th>
</tr>
</thead>
<tbody>
<tr>
<td>{b, kab}_{kAS}</td>
<td>{a, kab}_{kBS}</td>
<td>{_}_{kab}</td>
</tr>
</tbody>
</table>

\[ \text{out}(ca, \{b, kab\}_{kAS}).0 \]
| \[ \text{in}(cb, yb) \]
| \[ \text{let } (ya, yab) = \text{sdec}(yb, k_{BS}) \text{ in} \]
| \[ \text{if } ya = a \text{ then} \]
| \[ \text{out}(cb, \{ok\}_{yab}).0 \]|

\[ \text{in}(cs, zs) \]
| \[ \text{let } (zb, zab) = \text{sdec}(zs, k_{AS}) \text{ in} \]
| \[ \text{if } zb = b \text{ then} \]
| \[ \text{out}(cs, \{a, zab\}_{kBS}).0 \]|

Only 6 kinds of interleavings (instead of 60):
## Compressed semantics - Example

### Wide Mouthed frog

<table>
<thead>
<tr>
<th>Action</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice → Server</td>
<td>${b, kab}<em>{k</em>{AS}}$</td>
</tr>
<tr>
<td>Server → Bob</td>
<td>${a, kab}<em>{k</em>{BS}}$</td>
</tr>
<tr>
<td>Bob</td>
<td>${_}_{kab}$</td>
</tr>
</tbody>
</table>

**Partial order reduction**

\[
\begin{align*}
\text{out}(ca, \{b, kab\}_{k_{AS}}).0 = \text{in}(cb, yb) = sdec(yb, k_{BS}) \text{ in} & \\
\text{if } ya = a \text{ then} & \\
\text{out}(cb, \{ok\}_{yab}).0 & \\
\end{align*}
\]

\[
\begin{align*}
\text{in}(cs, zs) = sdec(zs, k_{AS}) \text{ in} & \\
\text{if } zb = b \text{ then} & \\
\text{out}(cs, \{a, zab\}_{k_{BS}}).0 & \\
\end{align*}
\]

Only 6 kinds of interleavings (instead of 60):
\[
\text{par.out}(ca, w_0).0
\]

---

**Lucca Hirschi**

**SEQUOIA: Partial Order Reduction for the applied $\pi$-calculus**

12 / 27
Wide Mouthed frog

Alice → Server : \{b, kab\}_{k_{AS}}
Server → Bob : \{a, kab\}_{k_{BS}}
Bob → : \{\_\}\_{kab}

\texttt{out}(ca, \{b, kab\}_{k_{AS}}).0 ; \texttt{in}(cb, yb)
let (ya, yab) = sdec(yb, k_{BS}) in
if ya = a then
  \texttt{out}(cb, \{ok\}_{yab}).0

\texttt{out}(ca, \{b, kab\}_{k_{AS}}).0 ; \texttt{in}(cb, yb)
let (ya, yab) = sdec(yb, k_{BS}) in
if ya = a then
  \texttt{out}(cb, \{ok\}_{yab}).0

\texttt{in}(cs, zs)
let (zb, zab) = sdec(zs, k_{AS}) in
if zb = b then
  \texttt{out}(cs, \{a, zab\}_{k_{BS}}).0

Only 6 kinds of interleavings (instead of 60):
\texttt{par.out}(ca, w_0).0.\texttt{in}(cb, Xb).\texttt{out}(cb, w_1).0.\texttt{in}(cs, Xs).\texttt{out}(cs, w_2).0
Compressed semantics - Example

Wide Moutthed frog

<table>
<thead>
<tr>
<th>Alice → Server</th>
<th>Server → Bob</th>
<th>Bob →</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>: {b, kab}_kAS</td>
<td>: {a, kab}_kBS</td>
</tr>
</tbody>
</table>

\[
\text{out}(ca, \{b, kab\}\_kAS).0
\]

; \text{in}(cb, yb)

\[
\text{let } (ya, yab) = sdec(yb, k_{BS}) \text{ in}
\]

if \(ya = a\) then

\[
\text{out}(cb, \{ok\}_{yab}).0
\]

; \text{in}(cs, zs)

\[
\text{let } (zb, zab) = sdec(zs, k_{AS}) \text{ in}
\]

if \(zb = b\) then

\[
\text{out}(cs, \{a, zab\}_{kBS}).0
\]

Only 6 kinds of interleavings (instead of 60):

\[
\text{par.out}(ca, w_0).0.\text{in}(cb, Xb).\text{out}(cb, w_1).0.\text{in}(cs, Xs).\text{out}(cs, w_2).0
\]

\[
\text{par.out}(ca, w_0).0.\text{in}(cs, Xs).\text{out}(cs, w_2).0.\text{in}(cb, Xb).\text{out}(cb, w_1).0
\]
Compressed semantics - Example

Wide Mouthed frog

Alice → Server : \{b, kab\}_{k_{AS}}
Server → Bob : \{a, kab\}_{k_{BS}}
Bob → : \{\_\}_{k_{ab}}

out(ca, \{b, kab\}_{k_{AS}}).0
; in(cb, yb)
let (ya, yab) = sdec(yb, k_{BS}) in
if ya = a then
  out(cb, \{ok\}_{yab}).0

; in(cs, zs)
let (zb, zab) = sdec(zs, k_{AS}) in
if zb = b then
  out(cs, \{a, zab\}_{k_{BS}}).0

Only 6 kinds of interleavings (instead of 60):
par.out(ca, w_0).0.in(cb, Xb).out(cb, w_1).0.in(cs, Xs).out(cs, w_2).0
par.out(ca, w_0).0.in(cs, Xs).out(cs, w_2).0.in(cb, Xb).out(cb, w_1).0
par.out(ca, w_0).0.in(cb, Xb).out(cb, w_1).0.in(cx, Xs).0
par.out(ca, w_0).0.in(cs, Xs).out(cs, w_2).0.in(cb, Xb).0
par.out(ca, w_0).0.in(cs, Xs).0
par.out(ca, w_0).0.in(cb, Xs).0
Compressed semantics - Replication

\[ Q = !^a_{c,n} \text{in}(c, x).\text{out}(c, \{< x, n >\}_k).0 \]

Compressed interleavings:
\[
\text{sess}(a, c_1).\text{in}(c_1, X_1).\text{out}(c_1, w_1).0 \\
\text{sess}(a, c_1).\text{in}(c_1, X_1).\text{out}(c_1, w_1).0.\text{sess}(a, c_2).\text{in}(c_2, X_2).\text{out}(c_2, w_2).0 \\
\ldots
\]
Lemma: soundness for reachability

Let $A$, $A'$, and $t$ be such that $A \xrightarrow{t_c} A'$. We have that $A \xrightarrow{t} A'$.

Easy.

Lemma: completeness for reachability

Let $A$, $A'$, and $t$ be such that $A \xrightarrow{t} A'$ is complete. There exists a trace $t_c$ that is a permutation of $t$ and $A \xrightarrow{t_c} A'$.

Sequential dependencies

We need to formalize sequential dependencies.

- add syntactical info. on process and produced actions
- *labels*: list of integers;
- denote the position of the current action in “the tree of parallel compositions”

Example

Labelled configuration:

\[ A = \{ [\text{in}(c, x). (\text{in}(c, x). \text{out}(c, m).0 | \text{in}(d, y). \text{out}(d, m').0^1) ] \} \]

Labelled trace:
\[ t = [\text{in}(c, x)]^1 \]
Sequential dependencies

We need to formalize sequential dependencies. 
- add syntactical info. on processes and produced actions
- *labels*: list of integers;
- denote the position of the current action in “the tree of parallel compositions”

Example

Labelled configuration:

\[ A = \{[\text{in}(c, x)]^1.[(\text{in}(c, x).\text{out}(c, m).0 \mid \text{in}(d, y).\text{out}(d, m').0)]^1\} \]

Labelled trace:

\[ t = [\text{in}(c, x)]^1[\text{par}]^1 \]
Sequential dependencies

We need to formalize sequential dependencies.

- add syntactical info. on process and produced actions
- *labels*: list of integers;
- denote the position of the current action in “the tree of parallel compositions”

Example

Labelled configuration:

\[ A = \left[ \text{in}(c, x).\text{out}(c, m).0 \right]^{1.1} \left[ \text{in}(d, y).\text{out}(d, m').0 \right]^{1.2} \]

Labelled trace:

\[ t = \left[ \text{in}(c, x) \right]^{1} [\text{par}^{1} \left[ \text{in}(c, x) \right]^{1.1} \left[ \text{out}(c, w_0) \right]^{1.1} \left[ \text{zero} \right]^{1.1} \left[ \text{in}(d, y) \right]^{1.2} \left[ \text{out}(d, w_1) \right]^{1.2} \left[ \text{zero} \right]^{1.2} \]
Swapping actions

Definition

\([\alpha]^{\ell}\) and \([\beta]^{\ell'}\) are *sequentially dependent* if \(\ell\) is a prefix of \(\ell'\) (or the converse).

Definition

\([\alpha]^{\ell}\) and \([\beta]^{\ell'}\) are *recipe dependent* if \(\{\alpha; \beta\} = \{\text{in}(c, M); \text{out}(d, w)\}\) with \(w \in \text{fv}(M)\).

We note \([\alpha]^{\ell} \parallel [\beta]^{\ell'}\) when they are recipe and sequentially independent.

Swapping Lemma

Consider a labelled configuration \(A\) and two actions \([\alpha]^{\ell} \parallel [\beta]^{\ell'}\). We have that

\[
A \xrightarrow{[\alpha]^{\ell} [\beta]^{\ell'}} A' \iff A \xrightarrow{[\beta]^{\ell'} [\alpha]^{\ell}} A'
\]

Key ingredient of the completeness proof.
Proof sketch of completeness

Let $A$, $A'$, and $t$ be such that $A \xrightarrow{t} A'$ is complete. There exists a trace $t_c$ such that $t_c$ is a permutation of $t$ and $A \xrightarrow{t} A'$.

Using the swapping Lemma we translate iteratively $A \xrightarrow{\text{tr}} A'$ into

$$A \xrightarrow{\ldots\text{tr}_{\text{pos}}\text{.tr}_{\text{neg}}\text{.tr}_{\text{pos}}\text{.tr}_{\text{neg}}\text{.tr}_{\text{pos}}\text{.tr}_{\text{neg}}\ldots} \xrightarrow{c} A'$$

$\leadsto$ Induction on the length of the derivation.
Proof sketch of completeness

Let $A$, $A'$, and $t$ be such that $A \xrightarrow{t} A'$ is complete. There exists a trace $t_c$ such that $t_c$ is a permutation of $t$ and $A \xrightarrow{t_c} A'$.

Using the swapping Lemma we translate iteratively $A \xrightarrow{t} A'$ into $A \xrightarrow{\ldots \text{tr}_{\text{pos}} \cdot \text{tr}_{\text{neg}} \cdot \text{tr}_{\text{pos}} \cdot \text{tr}_{\text{neg}} \cdot \text{tr}_{\text{pos}} \cdot \text{tr}_{\text{neg}} \ldots} A'$.  

$\rightsquigarrow$ Induction on the length of the derivation.

- If there is a negative $P_2 \in \mathcal{P}$. This $P_2$ performs $b_1$ at some point.
Proof sketch of completeness

Let \( A, A', \) and \( t \) be such that \( A \xrightarrow{t} A' \) is complete. There exists a trace \( t_c \) such that \( t_c \) is a permutation of \( t \) and \( A \xrightarrow{t_c} A' \).

Using the swapping Lemma we translate iteratively \( A \xrightarrow{tr} A' \) into

\[
A \xrightarrow{\ldots tr_{pos}.tr_{neg}.tr_{pos}.tr_{neg}.tr_{pos}.tr_{neg} \ldots} c A'
\]

\( \leadsto \) Induction on the length of the derivation.

- If there is a negative \( P_2 \in \mathcal{P} \). This \( P_2 \) performs \( b_1 \) at some point.
Proof sketch of completeness

Let \( A, A', \) and \( t \) be such that \( A \xrightarrow{t} A' \) is complete. There exists a trace \( t_c \) such that \( t_c \) is a permutation of \( t \) and \( A \xrightarrow{t_c} A' \).

Using the swapping Lemma we translate iteratively \( A \xrightarrow{tr} A' \) into

\[
A \xrightarrow{\ldots tr_{pos} \cdot tr_{neg} \cdot tr_{pos} \cdot tr_{neg} \cdot tr_{pos} \cdot tr_{neg} \ldots} c A'
\]

\( \leadsto \) Induction on the length of the derivation.

- Otherwise, \( A \) is positive. Only \textit{positive} actions leading to a \textit{negative} process \( P^- \).

\[
\begin{array}{c}
P_1 & a_1 \\
P_2 & b_1 \\
P_3 & c_1 \\
\end{array} \quad \quad | P^- \\
\begin{array}{c}
a_2 \\
a_3 \\
\end{array}
\]
Proof sketch of completeness

Let $A$, $A'$, and $t$ be such that $A \xrightarrow{t} A'$ is complete. There exists a trace $t_c$ such that $t_c$ is a permutation of $t$ and $A \xrightarrow{t_c} A'$.

Using the swapping Lemma we translate iteratively $A \xrightarrow{\text{tr}} A'$ into

$$A \xrightarrow{\ldots \text{tr}_{\text{pos}} \text{tr}_{\text{neg}} \text{tr}_{\text{pos}} \text{tr}_{\text{neg}} \text{tr}_{\text{pos}} \text{tr}_{\text{neg}} \ldots} c A'$$

$\leadsto$ Induction on the length of the derivation.

- Otherwise, $A$ is positive. Only positive actions leading to a negative process $P^-$.  

swapping lemma:

\[
\begin{align*}
P_1 & \quad a_1 & a_2 & a_3 \\
\downarrow P^- \downarrow & b_1 & \\
\downarrow P^- \downarrow & c_1 & \\
\end{align*}
\]
Results - Equivalence

To lift to equivalence we need to ensure that same swaps are possible on both sides

Strong symmetry Lemma

Let $A$ and $B$ be two action-deterministic configurations such that $A \approx B$ and $\text{skl}(A) = \text{skl}(B)$. For any execution

$$A \xrightarrow{[\alpha_1]^\ell_1} A_1 \xrightarrow{[\alpha_2]^\ell_2} \ldots \xrightarrow{[\alpha_n]^\ell_n} A_n$$

there exists an execution

$$B \xrightarrow{[\alpha_1]^\ell_1} B_1 \xrightarrow{[\alpha_2]^\ell_2} \ldots \xrightarrow{[\alpha_n]^\ell_n} B_n$$

such that $\Phi(A_i) \sim \Phi(B_i)$ and $\text{skl}(A_i) = \text{skl}(B_i)$ for any $1 \leq i \leq n$.

$$\text{skl}(\text{[in}(c, x).P)^{\text{lab}}) = \text{[in}_c]^{\text{lab}}, \text{skl}(\text{[!}_c, \overrightarrow{n}.P)^{\text{lab}}) = \text{[!}_a]^{\text{lab}}, \ldots$$
Results - Equivalence

To lift to equivalence we need to ensure that same swaps are possible on both sides

**Strong symmetry Lemma**

Let $A$ and $B$ be two action-deterministic configurations such that $A \approx B$ and $\text{skl}(A) = \text{skl}(B)$. For any execution

$$
A \xrightarrow{[\alpha_1]^{\ell_1}} A_1 \xrightarrow{[\alpha_2]^{\ell_2}} \ldots \xrightarrow{[\alpha_n]^{\ell_n}} A_n
$$

there exists an execution

$$
B \xrightarrow{[\alpha_1]^{\ell_1}} B_1 \xrightarrow{[\alpha_2]^{\ell_2}} \ldots \xrightarrow{[\alpha_n]^{\ell_n}} B_n
$$

such that $\Phi(A_i) \sim \Phi(B_i)$ and $\text{skl}(A_i) = \text{skl}(B_i)$ for any $1 \leq i \leq n$.

**Theorem:** $\approx_c \approx$

Let $A$ and $B$ be two action-deterministic configurations. $A \approx B$ if, and only if, $A \approx_c B$. 

Lucca Hirschi

SEQUOIA: Partial Order Reduction for the applied $\pi$-calculus
By building upon $\rightarrow_c, \approx_c$:

- compressed semantics produces *blocks* of actions of the form:

  $$b = t^+.t^-$$

- but we still need to make choices (which *positive* process, block?)

- some of them are redundant.
Intuitions

- compressed semantics produces blocks of actions of the form:
  \[ b = t^+ \cdot t^- \]

- but we still need to make choices (which positive process, block?)
- some of them are redundant.

\[ P = \mathit{in}(c_1, x_1) \cdot \mathit{out}(c_1, k_1) \cdot P_1 \mid \mathit{in}(c_2, x_2) \cdot \mathit{out}(c_2, k_2) \cdot P_2 \]
Intuitions

- compressed semantics produces *blocks* of actions of the form:
  \[ b = t^+.t^- \]

- but we still need to make *choices* (which *positive* process, block?)
- some of them are *redundant*.

\[
P = \text{in}(c_1, x_1).\text{out}(c_1, k_1).P_1 \mid \text{in}(c_2, x_2).\text{out}(c_2, k_2).P_2
\]

\[
\text{in}(c_1, X_1) \quad \text{in}(c_2, X_2)
\]

\[
\text{out}(c_1, w_1) \quad \text{out}(c_2, w_2)
\]

\[
\text{in}(c_2, X_2) \quad \text{in}(c_1, X_1)
\]

\[
\text{out}(c_2, w_2) \quad \text{out}(c_1, w_1)
\]

\[
\text{“}w_2 \in X_1\text{”}
\]

\[X_1\] must depend on \(w_2\).
Reduced semantics

We assume an arbitrary order \( \prec \) over blocks priority order.

**Semantics (informal)**

\[
A \xrightarrow{t} A' \quad A' \xrightarrow{b} A'' \quad \text{if } t \not\prec b
\]

\[
A \xrightarrow{t \cdot b} A'
\]

Informally, \( t \not\prec b \) means:

*there is no way to swap \( b \) towards the beginning of \( t \) before a block \( b_0 \not\succ b \) (even by modifying recipes)*
Reduced semantics

We assume an arbitrary order $≺$ over blocks priority order.

**Semantics (informal)**

\[
\begin{align*}
A & \xrightarrow{t\cdot r} A' \\
A' & \xrightarrow{b\cdot c} A'' \\
A & \xrightarrow{t\cdot b\cdot r} A'
\end{align*}
\]

if $t \triangleright b$

Informally, $t \triangleright b$ means:

*there is no way to swap $b$ towards the beginning of $t$ before a block $b_0 \triangleright b$ (even by modifying recipes)*

- $\text{in}(c_2, X_2) \cdot \text{out}(c_2, w_2) \triangleright \text{in}(c_1, X_1) \cdot \text{out}(c_1, w_1)$
- $X_1$ depends on $w_2$. 

Lucca Hirschi

SEQUOIA: Partial Order Reduction for the applied $\pi$-calculus
Reduced semantics

We assume an arbitrary order $\prec$ over blocks priority order.

Semantics (informal)

\[
\frac{A \xrightarrow{t} r A' A' \xrightarrow{b} c A''}{A \xrightarrow{t.b} r A'} \quad \text{if } t \not\succ b
\]

Informally, $t \not\succ b$ means:

*there is no way to swap $b$ towards the beginning of $t$ before a block $b_0 \succ b$ (even by modifying recipes)*

- $\text{in}(c_2, X_2).\text{out}(c_2, w_2) \not\succ \text{in}(c_1, X_1).\text{out}(c_1, w_1)$
  
  $X_1$ depends on $w_2$.

- $\text{in}(c_3, X_3).\text{out}(c_3, w_3).\text{in}(c_1, X_1).\text{out}(c_1, w_1) \not\succ \text{in}(c_2, X_2).\text{out}(c_2, w_2)$
  
  $X_2$ depends on either $w_1$ or $w_3$. 
Monoid of traces

Definition

Given a frame $\Phi$, the relation $\equiv_\Phi$ is the smallest equivalence over compressed traces such that:

- $t.b_1.b_2.t' \equiv_\Phi t.b_2.b_1.t'$ when $b_1 \parallel b_2$, and
- $t.b_1.t' \equiv_\Phi t.b_2.t'$ when $(b_1 =_E b_2)\Phi$.
Monoid of traces

**Definition**

Given a frame $\Phi$, the relation $\equiv_\Phi$ is the smallest equivalence over compressed traces such that:

- $t.b_1.b_2.t' \equiv_\Phi t.b_2.b_1.t'$ when $b_1 \parallel b_2$, and
- $t.b_1.t' \equiv_\Phi t.b_2.t'$ when $(b_1 =_E b_2)\Phi$.

**Lemma**

Let $A$ and $A'$ be two configurations such that $A \xrightarrow{t} c A'$. Then $A \xrightarrow{t'} c A'$ for any $t' \equiv_\Phi(A') \ t$.

**Goal:** explore on trace per equivalence class.
Results

Done: explore on trace per equivalence class.

t is $\Phi$-minimal if there is no $t' \prec_{\text{lex}} t$ such that $t \equiv_{\Phi} t'$

Lemma: completeness for reachability

If $A$ and $A'$ are and $A \xrightarrow{t} A'$ then $t$ is $\Phi(A')$-minimal if, and only if, $A \xrightarrow{t} r A'$. 
Done: explore on trace per equivalence class.

\[ t \text{ is } \Phi\text{-minimal if there is no } t' \prec_{\text{lex}} t \text{ such that } t \equiv_{\Phi} t' \]

**Lemma: completeness for reachability**

If \( A \) and \( A' \) are and \( A \xrightarrow{t} c A' \) then \( t \) is \( \Phi(A') \)-minimal if, and only if, \( A \xrightarrow{r} A' \).

- reduced semantics explores one trace per equivalence class
- with “swapping lemma” \( \rightsquigarrow \) completeness of reachability for \( \rightarrow_r \)
Results

Done: explore on trace per equivalence class.

\( t \) is \( \Phi \)-minimal if there is no \( t' \) such that \( t \equiv \Phi t' \)

Lemma: completeness for reachability

If \( A \) and \( A' \) are and \( A \xrightarrow{t} A' \) then \( t \) is \( \Phi(A') \)-minimal if, and only if, \( A \xrightarrow{t_r} A' \).

- reduced semantics explores one trace per equivalence class
- with "swapping lemma" \( \sim \) completeness of reachability for \( \rightarrow_r \)

Theorem

Let \( A \) and \( B \) be two action-deterministic configurations.

\[
A \simeq B \text{ if, and only if, } A \simeq_r B.
\]
Outline

1. Introduction
2. Model
3. Big Picture
4. Compression
5. Reduction
6. Conclusion
Implementations

Adapting well established techniques based on:

▶ symbolic semantics (abstract inputs);
▶ constraint solving procedures.

\( \text{tr}_\times b \) (availability) as a new type of constraints

Difficulties

▶ decide them exactly is too costly ...
Implementations

Adapting well established techniques based on:

- symbolic semantics (abstract inputs);
- constraint solving procedures.

tr\times b \text{ (availability)} as a new type of constraints

Difficulties

- decide them exactly is too costly ...
- \leadsto over-approximation;
- in a symmetrical way (otherwise false-attacks).
Implementations

Adapting well established techniques based on:

▶ symbolic semantics (abstract inputs);
▶ constraint solving procedures.

$\text{tr} \bowtie b$ (availability) as a new type of constraints

Difficulties

▶ decide them exactly is too costly ...  
▶ $\leadsto$ over-approximation;  
▶ in a symmetrical way (otherwise false-attacks).

We fully implemented our POR techniques (compression/reduction). They are now available in the main version of APTE:

```
github.com/APTE/
```
Benchmarks

Toy example ($\Pi_i(\text{in.out})$)

Instructions for reproduction:
www.lsv.ens-cachan.fr/~hirschi/apte_por
Conclusion

- New optimizations: compression and reduction;
- applied to trace equivalence checking;
- implementation in APTE.
Conclusion

- New optimizations: compression and reduction;
- applied to trace equivalence checking;
- implementation in APTE.

Future Work

1. drop action-deterministic assumption (?)
2. reducing search space:
   - study others redundancies \(\rightsquigarrow\) recognize symmetries ?
   - study constraint solving in more details
3. introduce interactivity into the verification process (sub-lemmas, annotations)
Conclusion

- New optimizations: compression and reduction;
- applied to trace equivalence checking;
- implementation in APTE.

Future Work

1. drop action-deterministic assumption (?)
2. reducing search space:
   - study others redundancies \(\leadsto\) recognize symmetries ?
   - study constraint solving in more details
3. introduce interactivity into the verification process (sub-lemmas, annotations)

Any question?
Compressed semantics - Definition

\[ \mathcal{P} \text{ is initial if } \forall P \in \mathcal{P}, \ P \text{ is positive or replicated.} \]

Semantics:
\( \mathcal{P} \text{ is initial if } \forall P \in \mathcal{P}, P \text{ is positive or replicated.} \)

Semantics:

\[
\begin{array}{c}
\text{START/IN} \\
\mathcal{P} \text{ is initial} \quad (P; \Phi) \xrightarrow{\text{in}(c,M)} (P'; \Phi) \\
(\mathcal{P} \cup \{P\}; \emptyset; \Phi) \xrightarrow{\text{foc(in}(c,M))} c (\mathcal{P} ; P'; \Phi) \\
(P; \Phi) \xrightarrow{\text{in}(c,M)} (P'; \Phi) \\
(\mathcal{P}; P; \Phi) \xrightarrow{\text{in}(c,M)} c (\mathcal{P} ; P'; \Phi)
\end{array}
\]
Compressed semantics - Definition

\( \mathcal{P} \) is **initial** if \( \forall P \in \mathcal{P}, P \) is **positive** or replicated.

**Semantics:**

\[
\begin{align*}
\text{START/IN} & \quad \mathcal{P} \text{ is initial} & \quad (P; \Phi) & \xrightarrow{\text{in}(c,M)} (P'; \Phi) \\
\quad & \quad (\mathcal{P} \cup \{P\}; \emptyset; \Phi) & \xrightarrow{\text{foc(in(c,M))}}_{c} (\mathcal{P}; P'; \Phi) \\
\quad & \quad (P; \Phi) & \xrightarrow{\text{in}(c,M)} (P'; \Phi) \\
\text{POS/IN} & \quad (\mathcal{P}; P; \Phi) & \xrightarrow{\text{in}(c,M)}_{c} (\mathcal{P}; P'; \Phi) \\
\text{RELEASE} & \quad (\mathcal{P}; P; \Phi) & \xrightarrow{\text{rel}}_{c} (\mathcal{P} \cup \{P\}; \emptyset; \Phi) \\
\text{NEG/\( \alpha \)} & \quad (\{P\}; \Phi) & \xrightarrow{\alpha} (\mathcal{P}'; \Phi') \\
\quad & \quad (\mathcal{P} \cup \{P\}; \emptyset; \Phi) & \xrightarrow{\alpha}_{c} (\mathcal{P} \cup \mathcal{P}'; \emptyset; \Phi') \\
\end{align*}
\]

\[\alpha \in \{\text{par, zero, out}(_{-}, _{-})\}\]
Reduced semantics

We assume an arbitrary order \( \prec \) over blocks (without recipes/messages): priority order.

Semantics

\[
A \xrightarrow{\epsilon} r A
\]

\[
A \xrightarrow{\text{tr}} (P; \emptyset; \Phi) (P; \emptyset; \Phi) \xrightarrow{b} A'
\]

if \( \text{tr} \triangleright b' \) for all \( b' \)

with \( (b' =_E b) \Phi \)

Availability

A block \( b \) is available after \( \text{tr} \), denoted \( \text{tr} \triangleright b \), if:

- either \( \text{tr} = \epsilon \)
- or \( \text{tr} = \text{tr}_0 . b_0 \) with \( \neg (b_0 \parallel b) \)
- or \( \text{tr} = \text{tr}_0 . b_0 \) with \( b_0 \parallel b \), \( b_0 \prec b \) and \( \text{tr}_0 \triangleright b \).
Benchmarks

Toy example ($\Pi_i(\text{in.out})$)

Wide Mouthed Frog

Instructions for reproduction: www.lsv.ens-cachan.fr/~hirschi/apte_por
Benchmarks

Toy example ($\Pi_i(\text{in.out})$)

Maximum number of parallel processes verifiable in 20 hours:

<table>
<thead>
<tr>
<th>Protocol</th>
<th>ref</th>
<th>comp</th>
<th>red</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yahalom (3-party)</td>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Needham Schroeder (3-party)</td>
<td>4</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>Private Authentication (2-party)</td>
<td>4</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>E-Passport PA (2-party)</td>
<td>4</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>Denning-Sacco (3-party)</td>
<td>5</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>Wide Mouthed Frog (3-party)</td>
<td>6</td>
<td>12</td>
<td>13</td>
</tr>
</tbody>
</table>

Instructions for reproduction:
www.lsv.ens-cachan.fr/~hirschi/apte_por