ProVerif, restrictions, equivalence... what could go wrong?

Alexandre Debant

Université de Lorraine, Inria, CNRS, Nancy, France

Pesto seminar
April 12th, 2024 - Nancy, France
Opening remarks

- this talk does not necessarily follow ProVerif notations

- what is written is not necessarily formally correct

- this talk is about ProVerif v2.05 (unless specific comment)
Modelling protocols

\[ P, Q := 0 \]
| new \( n \); \( P \)
| in\((c, x)\); \( P \)
| out\((c, u)\); \( P \)
| let \( u = \nu \) in \( P \) else \( Q \)
| insert \( tbl(u) \); \( P \)
| get \( tbl(x) \) such that \( \phi \) in \( P \) else \( Q \)
| \((P \mid Q)\)
| \(!P\)
| event \( e(u_1, \ldots, u_n) \); \( P \)

ProVerif before v2.02
Modelling protocols

\[ P, Q := 0 \]

| new \( n \); \( P \) |
| in\( (c, x) \); \( P \) |
| out\( (c, u) \); \( P \) |
| let \( u = v \) in \( P \) else \( Q \) |
| insert \( tbl(u) \); \( P \) |
| get \( tbl(x) \) suchthat \( \phi \) in \( P \) else \( Q \) |
| \( (P \mid Q) \) |
| \( !P \) |
| event \( e(u_1, \ldots, u_n) \); \( P \) |

ProVerif before v2.02

Restrictions:
\[ \rho := F_1 \& \cdots \& F_n \& \Rightarrow H \]

“Consider only traces that satisfy \( \rho \), i.e. \( tr \vdash \rho \)”

ProVerif since v2.02
Evoting: ballot weeding

Server =
  ! (  
    in(c, x);
    in(cell, x_{token});
    get BB(y) suchthat x = y in
      out(cell, x_{token}) (* ballot already accepted *)
  else
    insert BB(x);
    out(cell, x_{token});
    ...
Example

Evoting: ballot weeding

Server =
  ! (  
    in(c, x);
    in(cell, x_token);
    get BB(y) suchthat x = y in
      out(cell, x_token) (* ballot already accepted *)
    else
      insert BB(x);
      out(cell, x_token);
      ...  
  )

You may have troubles with else branches and cells ...
Evoting: ballot weeding

Server =
  ! (  
  in(c, x);  
in(cell, x_token);  
get BB(y) such that $x = y$ in  
  out(cell, x_token) (* ballot already accepted *)  
else  
  insert BB(x);  
  out(cell, x_token);  
  ...  
)

You may have troubles with else branches and cells ...

Server =
  ! (  
in(c, x);  
new st; event Inserted(st, x);  
insert BB(x);  
  ...  
)

Restriction:
  event(Inserted(st_1, x))  
  && event(Inserted(st_2, x)) ⇒ st_1 = st_2.
Example

Evoting: ballot weeding

\[
Server = ! ( \\
\quad \text{in}(c, x); \\
\quad \text{in}(\text{cell}, x_{\text{token}}); \\
\quad \text{get } BB(y) \text{ such that } x = y \text{ in} \\
\quad \quad \text{out}(\text{cell}, x_{\text{token}}) \quad (* \text{ballot already accepted} *) \\
\quad \text{else} \\
\quad \quad \text{insert } BB(x); \\
\quad \quad \text{out}(\text{cell}, x_{\text{token}}); \\
\quad \quad \ldots \\
) \\
\]

\[
\text{Restriction:} \\
\quad \text{event(Inserted}(st_1, x)) \quad \&\& \quad \text{event(Inserted}(st_2, x)) \Rightarrow st_1 = st_2. \\
\]

\[
\]

You may have troubles with else branches and cells ...
Other examples

- Ballot weeding in evoting protocols
  \[ \text{event(Inserted}(st_1, x) \&\& \text{event(Inserted}(st_2, x) \Rightarrow st_1 = st_2 \]

- Key updates / key revocations
  \[ \text{event(Use}(k_1) \&\& \text{event(Inserted}(k_2) \&\& \text{subterm}(k_1, k_2) \Rightarrow \text{false} \]

- Model protocol assumptions (e.g., audits)
  \[ \text{event(PublishedOnBB}(b)) \Rightarrow \phi(b) \]

- Easily bound the number of executions
  \[ \text{event(Iteration}(n)) \Rightarrow n < 2 \]

- Abstract e.g. arithmetic properties
  See [Cortier et. al. - CCS’21]

- ...
How does it work?
(simplified)

\[ \mathbb{C} \cup \{ R = H \rightarrow C \} \quad (\land_{i=1}^{n} F_i \Rightarrow \psi) \in \mathcal{R} \quad \text{For all } i, F_i \sigma \in H \]

\[ \frac{\mathbb{C} \cup \{ R = H \land \psi \sigma \rightarrow C \}}{} \]
How does it work?
(simplified)

\[
\begin{align*}
\mathbb{C} \cup \{ R = H \rightarrow C \} & \quad (\land_{i=1}^{n} F_i \Rightarrow \psi) \in \mathcal{R} & \text{For all } i, F_i \sigma \in H \\
\hline
\mathbb{C} \cup \{ R = H \land \psi \sigma \rightarrow C \}
\end{align*}
\]

It is just a matching!

If the clause is not instantiated enough (e.g. noselect) the restriction will not be applied!
Usual issues

Given the process $P := \text{event}(E_1); \text{event}(E_2); \text{event}(E_3)$
and the restriction $\rho := \text{event}(E_1) \Rightarrow \text{event}(E_2)$, is $\text{event}(E_3)$ reachable?
Usual issues

Given the process \( P := \text{event}(E_1); \text{event}(E_2); \text{event}(E_3) \)
and the restriction \( \rho := \text{event}(E_1) \Rightarrow \text{event}(E_2) \), is \( \text{event}(E_3) \) reachable?

No!
Restrictions have the same semantics as queries
Usual issues

Given the process $P := \text{event}(E1); \text{event}(E2); \text{event}(E3)$
and the restriction $\rho := \text{event}(E1) \Rightarrow \text{event}(E2)$,  
**is event($E3$) reachable?**

No!
Restrictions have the same semantics as queries

Given the process $P := (\text{event}(E1); \text{event}(E2)) | \text{event}(E3)$
and the restriction $\rho := \text{event}(E3) \Rightarrow \text{event}(E2)$,

is ProVerif able to prove $\rho' := \text{event}(E3) \Rightarrow \text{event}(E1)$?
Given the process and the restriction, is reachable?

\[
P : \mathcal{H} \mathcal{H} \mathcal{H}(E_1); \mathcal{H} \mathcal{H} \mathcal{H}(E_2); \mathcal{H} \mathcal{H} \mathcal{H}(E_3); \mathcal{H} \mathcal{H} \mathcal{H}(E_3) \Rightarrow \mathcal{H} \mathcal{H} \mathcal{H}(E_2) \Rightarrow \mathcal{H} \mathcal{H} \mathcal{H}(E_1)
\]

No!

Restrictions have the same semantics as queries

Given the process and the restriction, is ProVerif able to prove?

Derivation:

1. Event E3 may be executed at (3).

\[
\text{event}(E_3).
\]

2. By 1, event(E3).
The goal is reached, represented in the following fact:

\[
\text{event}(E_3).
\]

A more detailed output of the traces is available with

\[
\text{set traceDisplay = long}.
\]

\[
\text{event E3 at } (3) \text{ (goal)}
\]

The event E3 is executed at (3).

A trace has been found.

The attack trace does not satisfy the following restriction, declared at File "example4.pv", line 16, characters 13-35:

\[
\text{event(E3) \Rightarrow event(E2)}\]

RESULT event(E3) \Rightarrow event(E1) cannot be proved.
Usual issues

Given the process $P := \text{event}(E_1); \text{event}(E_2); \text{event}(E_3)$
and the restriction $\rho := \text{event}(E_1) \Rightarrow \text{event}(E_2)$, is $\text{event}(E_3)$ reachable?

No!
Restrictions have the same semantics as queries

Given the process $P := (\text{event}(E_1); \text{event}(E_2)) | \text{event}(E_3)$
and the restriction $\rho := \text{event}(E_3) \Rightarrow \text{event}(E_2)$,
is ProVerif able to prove $\rho' := \text{event}(E_3) \Rightarrow \text{event}(E_1)$?

No...
⇒ $\text{event}(E_3)$ apply $\rho$ $\Rightarrow \text{event}(E_2) \Rightarrow \text{event}(E_3)$ Not enough to conclude... 😞
Usual issues

Given the process $P := \text{event}(E1); \text{event}(E2); \text{event}(E3)$
and the restriction $\rho := \text{event}(E1) \Rightarrow \text{event}(E2)$, is $\text{event}(E3)$ reachable?

No!
Restrictions have the same semantics as queries

Given the process $P := (\text{event}(E1); \text{event}(E2)) \mid \text{event}(E3)$
and the restriction $\rho := \text{event}(E3) \Rightarrow \text{event}(E2)$,
is ProVerif able to prove $\rho' := \text{event}(E3) \Rightarrow \text{event}(E1)$?

No…
$\Rightarrow \text{event}(E3)$  $\xrightarrow{\text{apply } \rho}$  $\text{event}(E2) \Rightarrow \text{event}(E3)$

Not enough to conclude… 😢

You can use the development branch improve-scope-lemma to make it prove
What about equivalence properties?
• ProVerif proves equivalence of processes that differ only by terms

• ProVerif internally proves diff-equivalence

**Definition** - “A biprocess $P$ is in diff-equivalence if $\text{traces}(P)$ $\downarrow \uparrow$ i.e., for all traces of $P$, the first and the second projections progress in the same way.”

$$P[a_1, \ldots, a_n] \approx P[b_1, \ldots, b_n]$$

$$P[\text{diff}[a_1, b_1], \ldots, \text{diff}[a_n, b_n]] \uparrow \downarrow$$
Reminder

- ProVerif proves equivalence of processes that differ only by terms
- ProVerif internally proves diff-equivalence

**Definition** - “A biprocess \( P \) is in diff-equivalence if \( \text{traces}(P) \uparrow \downarrow \) i.e., for all traces of \( P \), the first and the second projections progress in the same way.”

\[
\begin{align*}
\text{let } x &= v \text{ in } P \text{ else } Q \mid \mathcal{P} &\rightarrow P \{ x \mapsto \text{diff}[M^L, M^R] \} \mid \mathcal{P} \\
&\quad \text{if } \text{fst}(v) \downarrow = M^L \text{ and } \text{snd}(v) \downarrow = M^R
\end{align*}
\]
Reminder

- ProVerif proves equivalence of processes that differ only by terms
- ProVerif internally proves diff-equivalence

**Definition** - “A biprocess $P$ is in diff-equivalence if $\text{traces}(P) \uparrow \downarrow$ i.e., for all traces of $P$, the first and the second projections progress in the same way.”

$P[a_1, \ldots, a_n] \approx P[b_1, \ldots, b_n]$

$\downarrow$

$P[\text{diff}[a_1, b_1], \ldots, \text{diff}[a_n, b_n]] \uparrow \downarrow$

$(\text{let } x = v \text{ in } P \text{ else } Q) \mid \mathcal{P} \rightarrow P\{x \mapsto \text{diff}[M^L, M^R]\} \mid \mathcal{P}$

if $\text{fst}(v) \downarrow = M^L$ and $\text{snd}(v) \downarrow = M^R$

$(\text{let } x = v \text{ in } P \text{ else } Q) \mid \mathcal{P} \rightarrow Q \mid \mathcal{P}$

if $\text{fst}(v) \downarrow = \text{fail}$ and $\text{snd}(v) \downarrow = \text{fail}$
Reminder

- ProVerif proves equivalence of processes that differ only by terms
- ProVerif internally proves diff-equivalence

**Definition** - “A biprocess $P$ is in diff-equivalence if $traces(P) \uparrow \downarrow$ i.e., for all traces of $P$, the first and the second projections progress in the same way.”

\[
P[a_1, \ldots, a_n] \approx P[b_1, \ldots, b_n]
\]

\[
P[diff[a_1, b_1], \ldots, diff[a_n, b_n]] \uparrow \downarrow
\]

\[
(P[x \mapsto \text{diff}[M^L, M^R]) | (let \ x = v \ in \ P \ else \ Q) \ | \ P \longrightarrow P\{x \mapsto \text{diff}[M^L, M^R]\} \ | \ P \\
\text{if } \text{fst}(v) \downarrow = M^L \text{ and } \text{snd}(v) \downarrow = M^R
\]

\[
(P[x \mapsto \text{diff}[M^L, M^R]) | (let \ x = v \ in \ P \ else \ Q) \ | \ P \longrightarrow Q \ | \ P \\
\text{if } \text{fst}(v) \downarrow = \text{fail} \text{ and } \text{snd}(v) \downarrow = \text{fail}
\]

\[
(P[x \mapsto u] | (in(c, x); P) \ | (out(c', u); Q) \ | \ P \longrightarrow P\{x \mapsto u\} \ | \ Q \ | \ P \\
\text{if } \text{fst}(c) = \text{fst}(c') \text{ and } \text{snd}(c) = \text{snd}(c')
\]

\[
\ldots
\]
Reminder

**Theorem** [Blanchet et. al. 2006]

Given a biprocess $P$, 

\[ \text{traces}(P) \downarrow \uparrow \Rightarrow \text{fst}(P) \approx \text{snd}(P) \]

where $\approx$ denotes the observational equivalence relation.
Reminder

**Theorem** [Blanchet et. al. 2006]

Given a biprocess $P$, \( \text{traces}(P) \downarrow \uparrow \Rightarrow \text{fst}(P) \approx \text{snd}(P) \)

where \( \approx \) denotes the observational equivalence relation.

```plaintext
adabent@macbook-pro-de-alexandre-2 proverif-examples ☰ proverif example1.pv

Biprocess \( \emptyset \) (that is, the initial process):
{
  (1)\text{new } n: \text{bitstring};
  (2)\text{new } m: \text{bitstring};
  (3)\text{out}(cpriv, \text{choice}(n,m))
} | {
  (4)\text{in}(cpriv, x: \text{bitstring});
  (5)\text{out}(cpub, x)
}

-- Observational equivalence in biprocess \( \emptyset \).
Translating the process into Horn clauses...
Termination warning: \( v \neq v_1 \land \text{attacker2}(v_2, v) \land \text{attacker2}(v_2, v_1) \rightarrow \text{bad} \)
Selecting 0
Termination warning: \( v \neq v_1 \land \text{attacker2}(v_2, v) \land \text{attacker2}(v_1, v_2) \rightarrow \text{bad} \)
Selecting 0
Completing...
Termination warning: \( v \neq v_1 \land \text{attacker2}(v_2, v) \land \text{attacker2}(v_2, v_1) \rightarrow \text{bad} \)
Selecting 0
Termination warning: \( v \neq v_1 \land \text{attacker2}(v_1, v_2) \land \text{attacker2}(v_1, v_2) \rightarrow \text{bad} \)
Selecting 0
RESULT Observational equivalence is true.

---------------------------------------------
Verification summary:
Observational equivalence is true.
---------------------------------------------
```
Equivalence with restrictions

- We can write restrictions, e.g.

$$\rho := \text{event}(E(\text{diff}[x^L, x^R], \text{diff}[y^L, y^R])) \Rightarrow x^L = y^L \land x^R = y^R$$
Equivalence with restrictions

- We can write restrictions, e.g.

\[ \rho := \text{event}(E(\text{diff}[x^L, x^R], \text{diff}[y^L, y^R])) \Rightarrow x^L = y^L \land x^R = y^R \]

\[ \rho' := \text{event}(E(x, y)) \Rightarrow x = y \neq \rho \]

\[ \rho' := \text{event}(E(x, y)) \Rightarrow x = y \equiv \text{event}(E(\text{diff}[x, x], \text{diff}[y, y])) \Rightarrow x = y \]
Equivalence with restrictions

- We can write restrictions, e.g.

\[ \rho := \text{event}(E(\text{diff}[x^L, x^R], \text{diff}[y^L, y^R])) \Rightarrow x^L = y^L \land x^R = y^R \]

\[ \rho' := \text{event}(E(x, y)) \Rightarrow x = y \equiv \text{event}(E(\text{diff}[x, x], \text{diff}[y, y])) \Rightarrow x = y \]

Always define restrictions with explicit `diff[·, ·]` operators!
We can write restrictions, e.g.

\[ \rho := \text{event}(E(\text{diff}[x^L, x^R], \text{diff}[y^L, y^R])) \Rightarrow x^L = y^L \land x^R = y^R \]

Always define restrictions with explicit \( \text{diff} [\cdot, \cdot] \) operators!

\[ \rho' := \text{event}(E(x, y)) \Rightarrow x = y \land \text{event}(E(\text{diff}[x,x], \text{diff}[y,y])) \Rightarrow x = y \]

**Definition** - A biprocess \( P \) is in diff-equivalence for the restrictions \( \mathcal{R} \), if \( \text{traces}_{\mathcal{R}}(P) \uparrow \) i.e., for all traces \( \text{tr} \) of \( P \) that satisfy \( \mathcal{R} \), \( \forall \rho \in \mathcal{R}, \text{tr} \vdash \rho \) the first and the second projections progress in the same way.
Relation with observational equivalence

**Definition** - Let $P^L, P^R$ be two processes and $R^L, R^R$ be two sets of restrictions. Observational equivalence is extended with restrictions as expected (i.e. considering only traces that satisfy restrictions 😊) and denoted $(P^L, R^L) \approx (P^R, R^R)$.
Relation with observational equivalence

Definition - Let $P^L, P^R$ be two processes and $\mathcal{R}^L, \mathcal{R}^R$ be two sets of restrictions. Observational equivalence is extended with restrictions as expected (i.e. considering only traces that satisfy restrictions 😊) and denoted $(P^L, \mathcal{R}^L) \approx (P^R, \mathcal{R}^R)$

New-theorem?

Given a biprocess $P$, and a set of restrictions $\mathcal{R}$,

$$\text{traces}_{\mathcal{R}}(P) \downarrow \uparrow \Rightarrow (\text{fst}(P), \text{fst}(\mathcal{R})) \approx (\text{snd}(P), \text{snd}(\mathcal{R})).$$
Relation with observational equivalence

**Definition** - Let $P^L, P^R$ be two processes and $\mathcal{R}^L, \mathcal{R}^R$ be two sets of restrictions. Observational equivalence is extended with restrictions as expected (i.e. considering only traces that satisfy restrictions 😄) and denoted $(P^L, \mathcal{R}^L) \approx (P^R, \mathcal{R}^R)$

**New-theorem?**

Given a biprocess $P$, and a set of restrictions $\mathcal{R}$,

$$\text{traces}\upharpoonright_{\mathcal{R}}(P) \downarrow \Rightarrow (\text{fst}(P), \text{fst}(\mathcal{R})) \approx (\text{snd}(P), \text{snd}(\mathcal{R})).$$

**FALSE**
**Definition** - Let $P^L$, $P^R$ be two processes and $\mathcal{R}^L$, $\mathcal{R}^R$ be two sets of restrictions. Observational equivalence is extended with restrictions as expected (i.e. considering only traces that satisfy restrictions 😊) and denoted $(P^L, \mathcal{R}^L) \approx (P^R, \mathcal{R}^R)$.
Why is it false?
Why is it false?

Strange restrictions

\[ \rho := \text{event}(E(\text{diff}[x^L, x^R])) \Rightarrow x^L = x^R \]
Strange restrictions

\[ \rho := \text{event}(E(\text{diff}[x^L, x^R])) \Rightarrow x^L = x^R \]  

\[ \text{fst}(\rho) \text{ is not properly defined!} \]
Why is it false?

Strange restrictions

\[ \rho := \text{event}(E(\text{diff}[x^L, x^R])) \Rightarrow x^L = x^R \]

\[ \times \] \text{fst(\rho) is not properly defined!}

A bi-restriction impact both sides of the equivalence
Why is it false?

Strange restrictions

\[ \rho := \text{event}(E(\text{diff}[x^L, x^R])) \Rightarrow x^L = x^R \]

\[ \text{fst}(\rho) \text{ is not properly defined!} \]

A bi-restriction impact both sides of the equivalence

\[ P = (\text{new } n; \text{ new } m; \text{ out}(cpriv1, \text{diff}[n, n]); \text{ out}(cpriv2, \text{diff}[n, m]); \text{ in}(cpriv1, x); \text{ in}(cpriv, y); \text{ event } E(x, y); \text{ out}(cpub, ok)) \]

Restriction: \( \rho := \text{event}(E(\text{diff}[x^L, x^R], \text{diff}[y^L, y^R])) \Rightarrow x^R = y^R \)
Why is it false?

Strange restrictions

\[ \rho := \text{event}(E(\text{diff}[x^L, x^R])) \Rightarrow x^L = x^R \]

\[ \times \text{fst}(\rho) \text{ is not properly defined!} \]

A bi-restriction impact both sides of the equivalence

\[
P = (\begin{array}{l}
\text{new } n; \text{ new } m; \\
\text{out}(cpriv1, \text{diff}[n, n]); \\
\text{out}(cpriv2, \text{diff}[n, m]); \\
\end{array}) \ | (\begin{array}{l}
\text{in}(cpriv1, x); \\
\text{in}(cpriv, y); \\
\text{event } E(x, y); \\
\text{out}(cpub, ok) \\
\end{array})
\]

Restriction: \[ \rho := \text{event}(E(\text{diff}[x^L, x^R], \text{diff}[y^L, y^R])) \Rightarrow x^R = y^R \]

\[
T := \text{out}(cpriv1, n).\text{in}(cpriv1, n). \\
\text{out}(cpriv2, n).\text{in}(cpriv2, n). \\
\text{event}(E(n, n)).\text{out}(cpub, ok)
\]

\[ T \in \text{traces}(\text{fst}(P)) \text{ and } T \vdash \text{true} = \text{fst}(\rho) \]

But \[ \text{event}(E(n, m)) \text{ cannot be executed in } \text{snd}(P) \text{ while satisfying } \text{snd}(\rho) \]
Why is it false?

Strange restrictions

\[ \rho := \text{event}(E(\text{diff}[x^L, x^R])) \Rightarrow x^L = x^R \]

\[ \text{\textbf{x}} \quad \text{fst(}\rho\text{)} \text{ is not properly defined!} \]

A bi-restriction impact both sides of the equivalence

\[ P = ( \]
\[ \text{new } n; \text{ new } m; \]
\[ \text{out}(cpriv1, \text{diff}[n, n]); \]
\[ \text{out}(cpriv2, \text{diff}[n, m]); \]
\[ ) \mid ( \]
\[ \text{in}(cpriv1, x); \]
\[ \text{in}(cpriv, y); \]
\[ \text{event } E(x, y); \]
\[ \text{out}(cpub, ok) \]
\[ ) \]

Restriction: \[ \rho := \text{event}(E(\text{diff}[x^L, x^R], \text{diff}[y^L, y^R])) \Rightarrow x^R = y^R \]

\[ T := \text{out}(cpriv1, n) . \text{in}(cpriv1, n) . \text{out}(cpriv2, n) . \text{in}(cpriv2, n) . \]
\[ \text{event}(E(n, n)) . \text{out}(cpub, ok) \]

\[ T \in \text{te} \]

\[ \text{snd(}T\text{)} \approx (\text{snd(}P\text{)}, \text{snd(}\rho\text{)}) \]

But \[ \text{event}(E(n, m)) \] cannot be executed in \[ \text{snd(}P\text{)} \] while satisfying \[ \text{snd(}\rho\text{)} \]
Why is it false?

Strange restrictions:

\[ \rho = \text{Restriction} \]

Restriction:

\[ \rho := \text{event} (E (\text{diff} [x^L, x^R], \text{diff} [y^L, y^R])) \Rightarrow x^R = y^R \]
What can I do now…?

I don’t know what I’m proving…
Solution 1

Trust yourself 👌

It’s the most often used technique… 🐘
Solution 2

Do a paper proof to justify each restriction...
Solution 3

Let ProVerif do the proof for you
Solution 3

Let ProVerif do the proof for you

**Methodology** - Given a biprocess $P$, and a restriction $\rho := F_1 \& \& \ldots \& \& F_n \Rightarrow H^L \& \& H^R$ such that:
- $\text{vars}(H^L) \subseteq \text{vars}($fst$(\rho))$ and $\text{vars}(H^R) \subseteq \text{vars}($snd$(\rho))$
- $\text{vars}($fst$(\rho)) \cap \text{vars}($snd$(\rho)) = \emptyset$

Let ProVerif prove that: for all $tr \in \text{traces}(P)$, $tr \vdash \overline{\text{fst}(\rho)}$ implies $tr \vdash \overline{\text{snd}(\rho)}$ and conversely.
Solution 3

Let ProVerif do the proof for you

Methodology - Given a biprocess $P$, and a restriction $\rho := F_1 \& \& \ldots \& \& F_n \Rightarrow H^L \& \& H^R$ such that:
- $\text{vars}(H^L) \subseteq \text{vars}(\text{fst}(\rho))$ and $\text{vars}(H^R) \subseteq \text{vars}(\text{snd}(\rho))$
- $\text{vars}(\text{fst}(\rho)) \cap \text{vars}(\text{snd}(\rho)) = \emptyset$

Let ProVerif prove that: for all $tr \in \text{traces}(P)$, $tr \vdash \text{fst}(\rho)$ implies $tr \vdash \text{snd}(\rho)$ and conversely.

Add $\text{diff}[, \cdot , \cdot]$ each time it is necessary with fresh variables on the right side.
Let ProVerif do the proof for you

**Methodology** - Given a biprocess $P$, and a restriction $\rho := F_1 \&\& \ldots \&\& F_n \Rightarrow H^L \&\& H^R$ such that:

- $\text{vars}(H^L) \subseteq \text{vars}(\text{fst}(\rho))$ and $\text{vars}(H^R) \subseteq \text{vars}(\text{snd}(\rho))$
- $\text{vars}(\text{fst}(\rho)) \cap \text{vars}(\text{snd}(\rho)) = \emptyset$

Let ProVerif prove that: for all $tr \in \text{traces}(P)$, $tr \vdash \overline{\text{fst}(\rho)}$ implies $tr \vdash \overline{\text{snd}(\rho)}$ and conversely.

Add $\text{diff}[, , ]$ each time it is necessary with fresh variables on the right side

**Example:** $\rho := \text{event}(E(\text{diff}[x^L, x^R], \text{diff}[y^L, y^R])) \Rightarrow x^L = y^L \&\& x^R = y^R$

$\overline{\text{fst}(\rho)} := \text{event}(E(\text{diff}[x^L, x_1], \text{diff}[y^L, x_2])) \Rightarrow x^L = y^L$

$\overline{\text{snd}(\rho)} := \text{event}(E(\text{diff}[x_1, x^R], \text{diff}[x_2, y^R])) \Rightarrow x^R = y^R$
Solution 3…
is not always possible…

The lemma talks about a unique trace…. in many cases you want to match the first side of a trace with the second side of another trace
Solution 3…
is not always possible…

The lemma talks about a unique trace…. in many cases you want to match the first side of a trace with the second side of another trace

\[ P := !\text{Reader} \mid !\text{new } k; \text{ !new } kk; \text{ insert } DB(\text{diff}[k, kk]); \text{ Tag(\text{diff}[k, kk])} \]
Solution 3... is not always possible...

The lemma talks about a unique trace.... in many cases you want to match the first side of a trace with the second side of another trace

$$P := \text{!}Reader \mid \text{!}\text{new } k; \text{!}\text{new } kk; \text{ insert } DB(\text{diff}[k, kk]); \text{ Tag(}\text{diff}[k, kk])$$

**Problem:** the key $k$ appears in many entries in $DB(\cdot)$,  
⇒ diff-equivalence does not hold...
The lemma talks about a unique trace. In many cases you want to match the first side of a trace with the second side of another trace.

Solution:

The Basic Hash protocol $P := !Reader | !new k; !new kk; \text{insert } DB(\text{diff}[k, kk]); \text{Tag(}\text{diff}[k, kk])$

**Problem:** the key $k$ appears in many entries in $DB(\cdot)$, $\Rightarrow$ diff-equivalence does not hold...

**Solution:** add a restriction to read the “good” entry when it exists.
Solution 3…
is not always possible…

The lemma talks about a unique trace…. in many cases you want to match the first side of a trace with the second side of another trace

\[ P := \text{!} \text{Reader} \mid \text{!} \text{new } k; \text{!} \text{new } kk; \text{ insert } DB(\text{diff}[k, kk]); \text{ Tag}(\text{diff}[k, kk]) \]

**Problem:** the key \( k \) appears in many entries in \( DB(\cdot) \),  
\[ \Rightarrow \text{diff-equivalence does not hold} \ldots \]

**Solution:** add a restriction to read the “good” entry when it exists

The previous lemma does not hold for traces using the “bad” entries
Solution 4
(ongoing work with Vincent and Itsaka)

Methodology
1. reinforce diff-equivalence to make it even stronger
2. adapt ProVerif procedure to make it sound w.r.t. this new definition
3. build upon Vincent and Itsaka’s approach [CSF’23] to discard false attacks
Solution 4
(ongoing work with Vincent and Itsaka)

Methodology
1. reinforce diff-equivalence to make it even stronger
2. adapt ProVerif procedure to make it sound w.r.t. this new definition
3. build upon Vincent and Itsaka’s approach [CSF’23] to discard false attacks

1. Reinforce diff-equivalence

Given a trace \( T \) and a well-formed restriction \( \rho \), \( T \downarrow \rho \) if \( T \uparrow \rho \) and for all \( T \rightarrow P \) we have:

\[
(T \rightarrow P) \vdash \text{fst}(\rho) \text{ if and only if } (T \rightarrow P) \vdash \text{snd}(\rho)
\]
2. Adapt ProVerif procedure - translation in “Horn” clauses

Given a process $P$, we note $C(P)$ the initial set of clauses generated by ProVerif.

Given a well-formed restriction $\rho := F_1 \land \ldots \land F_n \Rightarrow H_L \land H_R$, we define:

- $C^L_\rho = F_1 \land \ldots \land F_n \land H_L \land \neg H_R \Rightarrow \text{bad}$
- $C^R_\rho = F_1 \land \ldots \land F_n \land H_R \land \neg H_L \Rightarrow \text{bad}$

We define $C_\mathcal{R} = \{ C^X_\rho \mid \rho \in \mathcal{R}, X \in \{L, R\} \}$
2. Adapt ProVerif procedure - translation in “Horn” clauses

Given a process $P$, we note $C(P)$ the initial set of clauses generated by ProVerif.

Given a well-formed restriction $\rho := F_1 \& \& \cdots \& \& F_n \Rightarrow H^L \& \& H^R$, we define:

- $C^L_\rho = F_1 \& \& \cdots \& \& F_n \& \& H^L \& \& \neg H^R \Rightarrow \text{bad}$
- $C^R_\rho = F_1 \& \& \cdots \& \& F_n \& \& H^R \& \& \neg H^L \Rightarrow \text{bad}$

We define $C_\mathcal{R} = \{C^X_\rho \mid \rho \in \mathcal{R}, X \in \{L, R\}\}$

**Lemma** [soundness of the set of initial clauses]

Given a process $P$ and a set of well-formed restrictions $\mathcal{R}$, if $\neg P \uparrow \mathcal{R}$ then bad is derivable from $C(P) \cup C_\mathcal{R}$.
Solution 4
(ongoing work with Vincent and Itsaka)

2. Adapt ProVerif procedure - translation in “Horn” clauses

Given a process $P$, we note $\mathcal{C}(P)$ the initial set of clauses generated by ProVerif.

Given a well-formed restriction $\rho := F_1 \land \ldots \land F_n \Rightarrow H^L \land H^R$, we define:

- $C^L_\rho = F_1 \land \ldots \land F_n \land H^L \land \neg H^R \Rightarrow \text{bad}$
- $C^R_\rho = F_1 \land \ldots \land F_n \land H^R \land \neg H^L \Rightarrow \text{bad}$

We define $\mathcal{C}_{\mathcal{R}} = \{ C^X_\rho \mid \rho \in \mathcal{R}, X \in \{L, R\} \}$

**Lemma** [soundness of the set of initial clauses]

Given a process $P$ and a set of well-formed restrictions $\mathcal{R}$, if $\neg P \uparrow_{\mathcal{R}}$ then $\text{bad}$ is derivable from $\mathcal{C}(P) \cup \mathcal{C}_{\mathcal{R}}$.

Once this lemma is proved, the saturation is (almost) let unchanged, and thus its soundness proof too 😊
Solution 4
(ongoing work with Vincent and Itsaka)

3. Build upon Vincent and Itsaka’s approach [CSF’23] to discard false attacks

[Cheval & Rakotonirina - CSF’23] ==> ProVerif extension to (almost) prove session equivalence

Intuition:
- either the restriction is defined to discard some matchings (e.g. Basic Hash) and they are unnecessary to prove session equivalence
  ➔ Vincent&Itsaka extension will remove the newly reachable bad
- they are safe and bad should not be reachable
Solution 4
(ongoing work with Vincent and Itsaka)

3. Build upon Vincent and Itsaka’s approach [CSF’23] to discard false attacks

[Cheval & Rakotonirina - CSF’23] ==> ProVerif extension to (almost) prove session equivalence

Intuition:
- either the restriction is defined to discard some matchings (e.g. Basic Hash) and they are unnecessary to prove session equivalence
  - Vincent&Itsaka extension will remove the newly reachable bad
- they are safe and bad should not be reachable

TODO
- adapt Vincent&Itsaka extension (i.e. adapt all the proofs…)
- extend ProVerif (or find tricks) to support $\neg H_X$ in premise of a clause for any fact $H_X$
Conclusion

Be careful when you are using restrictions with equivalence queries…

It is not possible to think a bi-restriction as a restriction on the left side and a restriction on the right side.
Conclusion

Be careful when you are using restrictions with equivalence queries…

It is not possible to think a bi-restriction as a restriction on the left side and a restriction on the right side.

The manual of ProVerif and the long version of S&P’21 paper describe all the theory.

Everything is well-documented. Do not hesitate to open them when you’re not sure about what you’re proving.
Conclusion

Be careful when you are using restrictions with equivalence queries…

It is not possible to think a bi-restriction as a restriction on the left side and a restriction on the right side.

The manual of ProVerif and the long version of S&P’21 paper describe all the theory

Everything is well-documented. Do not hesitate to open them when you’re not sure about what you’re proving.

The improve-scope-lemma branch brings many new features

But part of them are under-documented…