Special topics in Logic and Security I

Master Year II, Sem. I, 2025-2026

Ioana Leuștean
FMI, UB

References

1 Cremers, C. J. F. (2006). Scyther : semantics and verification of security protocols Eindhoven: Technische Universiteit Eindhoven DOI: 10.6100/IR614943

2 Cremers C. and Mauw S. Operational Semantics and Verification of Security Protocols. Springer, 2012.

Verification

Verification

"In general, any nontrivial protocol has infinitely many possible behaviours (traces). However, from the perspective of property verification, many of these traces are just different interleavings or renamings of similar behaviours. To capture the concept of similar traces, we introduce trace patterns. A trace pattern, which represents a class of traces, is defined as a partially ordered set of symbolic events."

"A pattern represents a set of traces. This will allow us later to construct a verification algorithm for security properties that avoids considering all individual traces. To make the different ways in which the adversary can derive a term in execution traces explicit, we introduce adversary inference events."

"Our algorithm determines whether or not a particular pattern can occur in the traces of a protocol."

Cremers C. and Mauw S. Operational Semantics and Verification of Security Protocols. Springer, 2012.

The description of pattern events

$$
\begin{array}{lcl}
\mathit{AdvEvent} & ::= & \mathit{decr}(\{\!|\, \mathit{RunTerm} \,|\!\}_{\mathit{RunTerm}}) \mid \mathit{encr}(\{\!|\, \mathit{RunTerm} \,|\!\}_{\mathit{RunTerm}}) \mid \\
 & & \mid \mathit{app}(\mathit{Func}(\mathit{RunTerm}*)) \mid \mathit{init} \mid \mathit{know}(\mathit{RunTerm}), \\
\mathit{SendRecv} & ::= & \mathit{send} \mid \mathit{recv}, \\
\mathit{RolePos} & ::= & \mathit{Role}^{\#\mathit{RID}} \mid \mathit{Agent}, \\
\mathit{CommEvent} & ::= & \mathit{SendRecv}_{\mathit{label}}(\mathit{RolePos}, \mathit{RolePos}, \mathit{RunTerm})^{\#\mathit{RID}}, \\
\mathit{ClaimEvent} & ::= & \mathit{claim}_{\mathit{label}}(\mathit{RolePos}, \mathit{Claim}[, \mathit{RunTerm}]), \\
\mathit{AgEvent} & ::= & \mathit{CommEvent} \mid \mathit{ClaimEvent}, \\
\mathit{PatternEvent} & ::= & \mathit{AdvEvent} \mid \mathit{AgEvent}.
\end{array}
$$

- A pattern event is a partially instantiated run event.

PatternEvent

$$
\begin{array}{lcl}
\mathit{AdvEvent} & ::= & \mathit{decr}(\{\!|\, \mathit{RunTerm} \,|\!\}_{\mathit{RunTerm}}) \mid \mathit{encr}(\{\!|\, \mathit{RunTerm} \,|\!\}_{\mathit{RunTerm}}) \mid \\
 & & \mid \mathit{app}(\mathit{Func}(\mathit{RunTerm}*)) \mid \mathit{init} \mid \mathit{know}(\mathit{RunTerm}), \\
\mathit{SendRecv} & ::= & \mathit{send} \mid \mathit{recv}, \\
\mathit{RolePos} & ::= & \mathit{Role}^{\#\mathit{RID}} \mid \mathit{Agent}, \\
\mathit{CommEvent} & ::= & \mathit{SendRecv}_{\mathit{label}}(\mathit{RolePos}, \mathit{RolePos}, \mathit{RunTerm})^{\#\mathit{RID}}, \\
\mathit{ClaimEvent} & ::= & \mathit{claim}_{\mathit{label}}(\mathit{RolePos}, \mathit{Claim}[, \mathit{RunTerm}]), \\
\mathit{AgEvent} & ::= & \mathit{CommEvent} \mid \mathit{ClaimEvent}, \\
\mathit{PatternEvent} & ::= & \mathit{AdvEvent} \mid \mathit{AgEvent}.
\end{array}
$$

- The *init* event corresponds to adversary learning its initial knowledge.
- The $\mathit{know}(t)$ event corresponds to adversary learning t at some point in the trace.
- An adversary event is enabled if the adversary learns the necessary terms and has consequences that modifies the adversary knowledge. In order to model this interaction we define two functions from *PatternEvent* to *RunTerm*:
    - $\mathit{in}(e)$ is the set of terms that should be in adversary knowledge in order to enable the event
    - $\mathit{out}(e)$ is the set of terms that are added to the adversary knowledge after the execution of an event

The interaction between events and AKN

- $in(e)$ is the set of terms that should be in adversary knowledge in order to enable the event
- $out(e)$ is the set of terms that are added to the adversary knowledge after the execution of an event

e	$in(e)$	$out(e)$
$decr(\{\| t' \|\}_t)$	$\{\{\| t' \|\}_t\} \cup unpair(t^{-1})$	$unpair(t')$
$encr(\{\| t' \|\}_t)$	$unpair(t, t')$	$\{\{\| t' \|\}_t\}$
$app(f(t_1, \ldots, t_n))$	$unpair(t_0, \ldots, t_n)$	$\{f(t_1, \ldots, t_n)\}$
$know(t)$	$unpair(t)$	$\emptyset$
$init$	$\emptyset$	$\cup_{t \in AKN_0} unpair(t)$
$send_l(t)^{\#\theta}$	$\emptyset$	$unpair(t)$
$recv_l(t)^{\#\theta}$	$unpair(t)$	$\emptyset$
$claim_l(t, c, t')^{\#\theta}$	$\emptyset$	$\emptyset$

where $unpair(t, t') = \{t, t'\}$ (destructor for pairs)

Trace with adversary events

We consider the trace:
$tr = [$
$((1, \rho, \emptyset), create(i)),$
$((1, \rho, \emptyset), send(i, r, \{\!| m |\!\}_{sk(r)})),$
$((1, \rho, \emptyset\}), recv(r, i, \{\!| m, n_a |\!\}_{pk(r)}))]$

If $AKN_0 = \{n_a, pk(r)\}$ then a trace with adversary events is:

e	$in(e)$	$out(e)$
$init$	$\emptyset$	$AKN_0 = \{n_a, pk(r)\}$
$((1, \rho, \emptyset), create(i))$	$\emptyset$	$\emptyset$
$((1, \rho, \emptyset), send(i, r, \{\!| m |\!\}_{sk(r)}))$	$\emptyset$	$\{i, r, \{\!| m |\!\}_{sk(r)}\}$
$decr(\{\!| m |\!\}_{sk(r)})$	$\{\{\!| m |\!\}_{sk(r)}, pk(r)\}$	$\{m\}$
$encr(\{\!| m, n_a |\!\}_{pk(r)})$	$\{n_a, m, pk(r)\}$	$\{\{\!| m, n_a |\!\}_{pk(r)}\}$
$((1, \rho, \emptyset\}), recv(r, i, \{\!| m, n_a |\!\}_{pk(r)}))$	$\{r, i, \{\!| m, n_a |\!\}_{pk(r)}\}$	$\emptyset$

Pattern Variables *PVars*

$$\begin{array}{lcl}
\mathit{AdvEvent} & ::= & \mathit{decr}(\{\!|\ \mathit{RunTerm}\ |\!\}_{\mathit{RunTerm}}) \mid \mathit{encr}(\{\!|\ \mathit{RunTerm}\ |\!\}_{\mathit{RunTerm}}) \mid \\
 & & \mid \mathit{app}(\mathit{Func}(\mathit{RunTerm}*)) \mid \mathit{init} \mid \mathit{know}(\mathit{RunTerm}), \\
\mathit{SendRecv} & ::= & \mathit{send} \mid \mathit{recv}, \\
\mathit{RolePos} & ::= & \mathit{Role}^{\#\mathit{RID}} \mid \mathit{Agent}, \\
\mathit{CommEvent} & ::= & \mathit{SendRecv}_{\mathit{label}}(\mathit{RolePos}, \mathit{RolePos}, \mathit{RunTerm})^{\#\mathit{RID}}, \\
\mathit{ClaimEvent} & ::= & \mathit{claim}_{\mathit{label}}(\mathit{RolePos}, \mathit{Claim}[, \mathit{RunTerm}]), \\
\mathit{AgEvent} & ::= & \mathit{CommEvent} \mid \mathit{ClaimEvent}, \\
\mathit{PatternEvent} & ::= & \mathit{CommEvent} \mid \mathit{AgentEvent}.
\end{array}$$

$\mathit{PVars} = \{t^{\#\theta} \mid t \in \mathit{Var} \cup \mathit{Role}$ and $\theta \in \mathit{RID}\}$
(the set of variables and role terms that can occur in a pattern event)

- If $x_1, \ldots, x_n \in \mathit{PVars}$ and $t_1, \ldots, t_n \in \mathit{RunTerm}$ then we denote a *substitution* $\phi : \mathit{PVars} \to \mathit{RunTerm}$ by $\phi = [t_1, \ldots, t_n / x_1, \ldots, x_n]$
- A substitution ϕ is a *unifier* for t and t' if $\phi(t) = \phi(t')$.

Patterns for a protocol P

$$Protocol = Role \rightharpoondown RoleSpec$$

E a finite set of pattern events
$\rightarrow \subseteq E \times E$ a relation labeled with elements from $RunTerm \cup \{\lambda\}$
$(E, \rightarrow)$ is a *pattern* for P if and only if
for any $e, e' \in E$, $t \in RunTerm$, $re, re' \in RoleEvent$ and $\theta \in RID$:

(p1) Message causality: if $e \xrightarrow{t} e'$ then $t \in out(e) \cap in(e')$

(p2) (agent events are from P) if $e \in AgEvent$ then there exists $(M, s_1 \cdots s_n) \in RoleSpec$ such that $e = \phi(s_i)^{\#\theta}$ for some initial knowledge M, role event s_i and substitution ϕ;

(p3) Role events are unique per run: if $e = \phi(re)^{\#\theta}$ and $e' = \phi'(re)^{\#\theta}$ then $e = e'$ for any ϕ, ϕ' substitutions;

(p4) Runs are prefix-closed with respect to the agent events: if $e = \phi(re)^{\#\theta}$ and $re' <_R re$ the $\phi(re')^{\#\theta} \rightarrow^* e$ for any role R and substitution ϕ.

Example: pattern for the Needham-Schroeder protocol

$$\begin{tikzcd}
recv_1(i^{\#1}, r^{\#1}, rt_1)^{\#2} \arrow[d, "\lambda"] & send_1(i^{\#1}, r^{\#1}, rt_1)^{\#1} \arrow[l, "\mathbf{rt_1}"'] \arrow[d, "\lambda"] \\
send_2(r^{\#1}, i^{\#1}, rt_2)^{\#2} \arrow[r, "\mathbf{rt_2}"] & recv_2(r^{\#1}, i^{\#1}, rt_2)^{\#1} \arrow[d, "\lambda"] \\
 & send_3(i^{\#1}, r^{\#1}, rt_3)^{\#1}
\end{tikzcd}$$

where

$rt_1 = \{\!| \, n_i^{\#1}, i^{\#1} \, |\!\}_{pk(r^{\#1})}$

$rt_2 = \{\!| \, n_i^{\#1}, n_r^{\#2} \, |\!\}_{pk(i^{\#1})}$

$rt_3 = \{\!| \, n_r^{\#2} \, |\!\}_{pk(r^{\#1})}$

Traces as patterns

- We can associate a pattern with any trace. The pattern edges are defined by the trace order and are labeled with λ.

For $e \in \mathit{RunEvent}$ and $\mathit{inst} = (\theta, \rho, \sigma) \in \mathit{Inst}$ we define

$$\langle \mathit{inst} \rangle(e) = \begin{cases} \mathit{send}_l(\mathit{inst}((r, r', m)))^{\#\theta} & \text{iff } e = \mathit{send}_l(r, r', m), \\ \mathit{recv}_l(\mathit{inst}((r, r', m)))^{\#\theta} & \text{iff } e = \mathit{recv}_l(r, r', m), \\ \mathit{claim}_l(\mathit{inst}(r), c, \mathit{inst}(m))^{\#\theta} & \text{iff } e = \mathit{claim}_l(r, c, m). \end{cases}$$

The function $\mathit{IT} : \mathit{RunEvent}^* \rightarrow \mathit{PatternEvent}^*$ defined by $\mathit{IT}([]) = []$ and

$$\langle \mathit{inst} \rangle(e) = \begin{cases} [\langle \mathit{inst} \rangle(e)] \cdot \mathit{IT}(\mathit{tr}) & \text{if } e \text{ is not a } \mathit{create} \text{ event}, \\ \mathit{IT}(\mathit{tr}) & \mathit{otherwise}, \end{cases}$$

returns the pattern associated with a trace.

Traces of a protocol and a pattern

P a protocol, $(E, \rightarrow)$ a pattern

We define $\mathit{traces}(P, (E, \rightarrow))$ as the set of all traces tr of P with the following properties:

- for any $e, e' \in \mathit{AgEvent}$, if $e \rightarrow^* e'$ then $\phi(\zeta(e)) \leq_{IT(tr)} \phi(\zeta(e'))$
- if $\mathit{know}(t) \in E$ then $\mathit{AKN}(tr) \vdash \phi(\zeta(t))$

where $\phi : \mathit{PVars} \rightarrow \mathit{RunTerm}$ is a well-typed substitution, $\zeta : \mathit{RID} \rightarrow \mathit{RID}$ is a bijection.

If $tr \in \mathit{traces}(P, (E, \rightarrow))$ we say that *tr is a trace of* $(E, \rightarrow)$ (or *trace tr exhibits the pattern* $(E, \rightarrow)$).

Security property are defined as *pattern problem*:
if *sec* is a security property, and *pt* is the associated pattern, then in order to prove that *sec* holds, we verify that $\mathit{traces}(P, pt) = \emptyset$.

Example: trace for the Needham-Schroeder protocol

$((1, \rho, \emptyset), \mathit{create}(i))$

$((1, \rho, \emptyset), \mathit{send}_1(i, r, \{\!| \, ni, i \, |\!\}_{pk(r)}))$

$((2, \rho, \emptyset), \mathit{create}(r))$

$((2, \rho, \{W \mapsto ni^{\#1}\}), \mathit{recv}_1(i, r, \{\!| \, W, i \, |\!\}_{pk(r)}))$

$((2, \rho, \{W \mapsto ni^{\#1}\}), \mathit{send}_2(r, i, \{\!| \, W, nr \, |\!\}_{pk(i)}))$

$((1, \rho, \{V \mapsto nr^{\#2}\}), \mathit{recv}_2(r, i, \{\!| \, ni, V \, |\!\}_{pk(i)}))$

$((1, \rho, \{V \mapsto nr^{\#2}\}), \mathit{send}_3(i, r, \{\!| \, V \, |\!\}_{pk(r)}))$

$((1, \rho, \{V \mapsto nr^{\#2}\}), \mathit{claim}_4(i, \mathit{synch}))$

$((2, \rho, \emptyset), \mathit{recv}_3(i, r, \{\!| \, nr \, |\!\}_{pk(r)}))$

$((2, \rho, \emptyset), \mathit{claim}_5(r, \mathit{synch}))$

$$NS : \mathit{Role} \rightharpoonup \mathit{RoleSpec}$$

$$\begin{aligned}
NS(i) = \; & (\{i, r, ni, sk(i), pk(i), pk(r)\}, \\
& [\mathit{send}_1(i, r, \{\!| \, ni, i \, |\!\}_{pk(r)}), \\
& \mathit{recv}_2(r, i, \{\!| \, ni, V \, |\!\}_{pk(i)}), \\
& \mathit{send}_3(i, r, \{\!| \, V \, |\!\}_{pk(r)}), \\
& \mathit{claim}_4(i, \mathit{synch})])
\end{aligned}$$

$$\begin{aligned}
NS(r) = \; & (\{i, r, nr, sk(r), pk(r), pk(i)\}, \\
& [\mathit{recv}_1(i, r, \{\!| \, W, i \, |\!\}_{pk(r)}), \\
& \mathit{send}_2(r, i, \{\!| \, W, nr \, |\!\}_{pk(i)}), \\
& \mathit{recv}_3(i, r, \{\!| \, nr \, |\!\}_{pk(r)}), \\
& \mathit{claim}_5(r, \mathit{synch})])
\end{aligned}$$

Example: pattern for the Needham-Schroeder protocol

$((1, \rho, \emptyset), \mathit{create}(i))$

$((1, \rho, \emptyset), \mathit{send}_1(i, r, \{\!|\, ni, i \,|\!\}_{pk(r)}))$

$((2, \rho, \emptyset), \mathit{create}(r))$

$((2, \rho, \{W \mapsto ni^{\#1}\}), \mathit{recv}_1(i, r, \{\!|\, W, i \,|\!\}_{pk(r)}))$

$((2, \rho, \{W \mapsto ni^{\#1}\}), \mathit{send}_2(r, i, \{\!|\, W, nr \,|\!\}_{pk(i)}))$

$((1, \rho, \{V \mapsto nr^{\#2}\}), \mathit{recv}_2(r, i, \{\!|\, ni, V \,|\!\}_{pk(i)}))$

$((1, \rho, \{V \mapsto nr^{\#2}\}), \mathit{send}_3(i, r, \{\!|\, V \,|\!\}_{pk(r)}))$

$((1, \rho, \{V \mapsto nr^{\#2}\}), \mathit{claim}_4(i, \mathit{synch}))$

$((2, \rho, \emptyset), \mathit{recv}_3(i, r, \{\!|\, nr \,|\!\}_{pk(r)}))$

$((2, \rho, \emptyset), \mathit{claim}_5(r, \mathit{synch}))$

$$\begin{tikzcd}
\mathit{recv}_1(i^{\#0}, r^{\#0}, \{\!|\, W^{\#0}, i^{\#0} \,|\!\}_{pk(r^{\#0})})^{\#0}, \arrow[d, "\lambda"] \\
\mathit{send}_2(r^{\#0}, i^{\#0}, \{\!|\, W^{\#0}, nr^{\#0} \,|\!\}_{pk(i^{\#0})})^{\#0}, \arrow[d, "\lambda"] \\
\mathit{recv}_3(i^{\#0}, r^{\#0}, \{\!|\, nr^{\#0} \,|\!\}_{pk(r^{\#0})})^{\#0}
\end{tikzcd}$$

Secrecy as pattern problem

Security property are defined as *pattern problem*: if *sec* is a security property, and *pt* is the associated pattern, then in order to prove that *sec* holds, we verify that $traces(P, pt) = \emptyset$.

- Secrecy as pattern problem:

  Let P be a protocol,R a role, $P(R) = (M, s)$, $s_i = claim_l(R, secret, rt)$, $\theta \in RID$, $inst = (\theta, \rho, \emptyset)$ where $\rho(r) = r^{\#\theta}$ for any $r \in Role$. We define the pattern $pt = (E, \rightarrow)$ as follows:

  - $E = \{\langle inst \rangle(s_0), \ldots, \langle inst \rangle(s_i)\} \cup \{know(\langle inst \rangle(rt))\}$
  - $\rightarrow$ is: $s_0 \overset{\lambda}{\rightarrow} s_1 \overset{\lambda}{\rightarrow} \cdots \overset{\lambda}{\rightarrow} s_i$
  - $type(r^{\#\theta}) = Agent_H$ (honest) for any role r

  The *secrecy* claim holds if and only if $traces(P, pt) = \emptyset$.

NS protocol: the attack trace for the secrecy claim

$$
\begin{aligned}
&[((1,\rho_1,\emptyset), \mathit{create}(i))\ ,\\
&((1,\rho_1,\emptyset), \mathit{send}_1(i,r,\{\!|\ ni, i\ |\!\}_{pk(r)})),\\
&((2,\rho_2,\emptyset), \mathit{create}(r)),\\
&((2,\rho_2,\{W \mapsto ni^{\#1}\}), \mathit{recv}_1(i,r,\{\!|\ W, i\ |\!\}_{pk(r)})),\\
&((2,\rho_2,\{W \mapsto ni^{\#1}\}), \mathit{send}_2(r,i,\{\!|\ W, nr\ |\!\}_{pk(i)})),\\
&((1,\rho_1,\{V \mapsto nr^{\#2}\}), \mathit{recv}_2(r,i,\{\!|\ ni, V\ |\!\}_{pk(i)})),\\
&((1,\rho_1,\{V \mapsto nr^{\#2}\}), \mathit{send}_3(i,r,\{\!|\ V\ |\!\}_{pk(r)})),\\
&((2,\rho_2,\{W \mapsto ni^{\#1}\}), \mathit{recv}_3(i,r,\{\!|\ nr\ |\!\}_{pk(r)})),\\
&((2,\rho_2,\{W \mapsto ni^{\#1}\}), \mathit{claim}_5(r,\mathit{secret},nr))]
\end{aligned}
$$

$$NS : Role \rightharpoonup RoleSpec$$

$$
\begin{aligned}
NS(i) = \quad &(\{i, r, ni, sk(i), pk(i), pk(r)\},\\
&[\mathit{send}_1(i, r, \{\!|\ ni, i\ |\!\}_{pk(r)}),\\
&\mathit{recv}_2(r, i, \{\!|\ ni, V\ |\!\}_{pk(i)}),\\
&\mathit{send}_3(i, r, \{\!|\ V\ |\!\}_{pk(r)}),\\
&\mathit{claim}_4(i, \mathit{synch})])
\end{aligned}
$$

$$
\begin{aligned}
NS(r) = \quad &(\{i, r, nr, sk(r), pk(r), pk(i)\},\\
&[\mathit{recv}_1(i, r, \{\!|\ W, i\ |\!\}_{pk(r)}),\\
&\mathit{send}_2(r, i, \{\!|\ W, nr\ |\!\}_{pk(i)}),\\
&\mathit{recv}_3(i, r, \{\!|\ nr\ |\!\}_{pk(r)}),\\
&\mathit{claim}_5(r, \mathit{synch})])
\end{aligned}
$$

NS protocol: the attack pattern for the secrecy claim

$[((1,\rho_1,\emptyset), create(i))\ ,$
$((1,\rho_1,\emptyset), send_1(i,r,\{\!|\ ni, i\ |\!\}_{pk(r)})),$
$((2,\rho_2,\emptyset), create(r)),$
$((2,\rho_2,\{W \mapsto ni^{\#1}\}), recv_1(i,r,\{\!|\ W, i\ |\!\}_{pk(r)})),$
$((2,\rho_2,\{W \mapsto ni^{\#1}\}), send_2(r,i,\{\!|\ W, nr\ |\!\}_{pk(i)})),$
$((1,\rho_1,\{V \mapsto nr^{\#2}\}), recv_2(r,i,\{\!|\ ni, V\ |\!\}_{pk(i)})),$
$((1,\rho_1,\{V \mapsto nr^{\#2}\}), send_3(i,r,\{\!|\ V\ |\!\}_{pk(r)})),$
$((2,\rho_2,\{W \mapsto ni^{\#1}\}), recv_3(i,r,\{\!|\ nr\ |\!\}_{pk(r)})),$
$((2,\rho_2,\{W \mapsto ni^{\#1}\}), claim_4(r, secret, W))]$

$$know(W^{\#0})$$

$$\begin{tikzcd}
recv_1(i^{\#0}, r^{\#0}, \{\!|\ W^{\#0}, i^{\#0}\ |\!\}_{pk(r^{\#0})})^{\#0} \arrow[d, "\lambda"] \\
send_2(r^{\#0}, i^{\#0}, \{\!|\ W^{\#0}, nr^{\#0}\ |\!\}_{pk(i^{\#0})})^{\#0} \arrow[d, "\lambda"] \\
recv_3(i^{\#0}, r^{\#0}, \{\!|\ nr^{\#0}\ |\!\}_{pk(r^{\#0})})^{\#0} \arrow[d, "\lambda"] \\
claim_4(i^{\#0}, r^{\#0}, W^{\#0})^{\#0}
\end{tikzcd}$$

$$type(i^{\#0}) = type(r^{\#0}) = Agent_H$$

Realisable patterns

P a protocol, $(E, \to)$ a pattern

- We say that a pattern (E, to) is *realisable* if for any $e \in E$ and for any $t \in (in(e) \setminus PVars)$ there is some $e' \in E$ such that $e' \to^* e$ and $t \in out(e')$. In addition, we require that $\to$ is not cyclic and that the substitution from (p2) is well-typed.

  The above definition says that if some knowledge is needed to trigger an adversary event from the pattern, then there are some preceding events that made the information available, with the exception of *PVars*, which are excluded since the adversary can always generate new terms.

- The trace of a realisable pattern: Let $(E, \to)$ be a realisable pattern and $\leq$ a total order such that $\to \subseteq \leq$. Let $\phi : PVars \to RunTerm$ such that $\phi(PVars) \subseteq \{m \mid AKN_0 \vdash m\}$. Then there exists a trace $tr \in traces(P, (E, \to)))$ such that

$$IT(tr) = \phi((E, \leq))$$

Example

Non-realisable patterns

$$\begin{tikzcd} recv_1(i^{\#0}, r^{\#0}, \{\!|\, W^{\#0}, i^{\#0} \,|\!\}_{pk(r^{\#0})})^{\#0}, \arrow[d, "\lambda"] \\ send_2(r^{\#0}, i^{\#0}, \{\!|\, W^{\#0}, nr^{\#0} \,|\!\}_{pk(i^{\#0})})^{\#0}, \arrow[d, "\lambda"] \\ recv_3(i^{\#0}, r^{\#0}, \{\!|\, nr^{\#0} \,|\!\}_{pk(r^{\#0})})^{\#0} \end{tikzcd}$$

$know(W^{\#0})$

$$\begin{tikzcd} recv_1(i^{\#0}, r^{\#0}, \{\!|\, W^{\#0}, i^{\#0} \,|\!\}_{pk(r^{\#0})})^{\#0} \arrow[d, "\lambda"] \\ send_2(r^{\#0}, i^{\#0}, \{\!|\, W^{\#0}, nr^{\#0} \,|\!\}_{pk(i^{\#0})})^{\#0} \arrow[d, "\lambda"] \\ recv_3(i^{\#0}, r^{\#0}, \{\!|\, nr^{\#0} \,|\!\}_{pk(r^{\#0})})^{\#0} \arrow[d, "\lambda"] \\ claim_4(i^{\#0}, r^{\#0}, W^{\#0})^{\#0} \end{tikzcd}$$

$type(i^{\#0}) = type(r^{\#0}) = Agent_H$

- the patterns are *not realisable* because there are no incoming edges for the encryptions in the first receive event

Example

Realisable pattern for the Needham-Schroeder protocol

$$\begin{tikzcd}
recv_1(i^{\#1}, r^{\#1}, rt_1)^{\#2} \arrow[d, "\lambda"] & send_1(i^{\#1}, r^{\#1}, rt_1)^{\#1} \arrow[l, "\mathbf{rt_1}"'] \arrow[d, "\lambda"] \\
send_2(r^{\#1}, i^{\#1}, rt_2)^{\#2} \arrow[r, "\mathbf{rt_2}"] & recv_2(r^{\#1}, i^{\#1}, rt_2)^{\#1} \arrow[d, "\lambda"] \\
 & send_3(i^{\#1}, r^{\#1}, rt_3)^{\#1}
\end{tikzcd}$$

where

$rt_1 = \{\!| \; n_i^{\#1}, i^{\#1} \; |\!\}_{pk(r^{\#1})}$

$rt_2 = \{\!| \; n_i^{\#1}, n_r^{\#2} \; |\!\}_{pk(i^{\#1})}$

$rt_3 = \{\!| \; n_r^{\#2} \; |\!\}_{pk(r^{\#1})}$

Algorithm overview for a protocol P

- Define a pattern $pt = (E, \rightarrow)$ representing a set of traces.
  - Pattern representing the set of traces violating a security property (secrecy).
  - Patterns representing al the executions of a specific role (defined using the events of the role for honest agents and the *init* adversary event).

- If pt is not realisable, then apply an algorithm that refines the pattern, providing a pattern pt' such that $traces(P, pt') \subseteq traces(P, pt)$.
  - Patterns can be refined by adding events, adding edges, or performing well-typed substitutions.

- The algorithm returns a set of realisable patterns, that represents the same set of traces as pt.
  - If pt represents a security property (such as secrecy) and the set of realisable patterns returned by the algorithm is nonempty, then the property is violated.
  - If pt represents all the possible executions of a role, the security properties are verified for each relisable pattern.

Operational Semantics

1 Cremers, C. J. F. (2006). Scyther : semantics and verification of security protocols Eindhoven: Technische Universiteit Eindhoven DOI: 10.6100/IR614943

2 Cremers C. and Mauw S. Operational Semantics and Verification of Security Protocols. Springer, 2012.