2.6 Operational semantics

The process algebra is a quotient structure. We consider as identical processes that are structurally equivalent. The structural equivalence $\equiv$ is defined by:

$$\begin{array}{rcll}
P\|Q & \equiv & Q\|P & \\
P\|(Q\|R) & \equiv & (P\|Q)\|R & \\
!P & \equiv & P\|!P & \\
\nu n.\ P & \equiv & \nu n'.\ P\{n \mapsto n'\} & \text{if } n' \notin \mathsf{fn}(P) \cup \mathsf{bn}(P) \\
(\nu n.P)\|Q & \equiv & \nu n.\ (P\|Q) & \text{if } n \notin \mathsf{fn}(Q) \\
P\|\mathbf{0} & \equiv & P & \\
\nu n.\ \nu n'.\ P & \equiv & \nu n'.\ \nu n.\ P &
\end{array}$$

Communications are synchronous, but (as long as there is no private channel) they always occur with the environment. We sketch here a labeled transition semantics. For, we consider generalized processes, which consist of a frame and a process. More precisely, a general process is an expression

$$\nu\overline{n}.(\{x_1 \mapsto t_1;\ \cdots\ ; x_k \mapsto t_k\}\|P)$$

where P is a process and $t_1, \ldots, t_k$ are terms.

$$\frac{P \equiv Q \quad P\ \rightarrow\ P'}{Q\ \rightarrow\ P'}$$

$$\frac{}{\nu\overline{n}.(\sigma\|\mathsf{out}(c,s).P\|Q \rightarrow \sigma \uplus \{x \mapsto s\}\|P\|Q)} \quad \text{if } x \notin \mathsf{Dom}(\sigma)$$

$$\frac{}{\nu\overline{n}.\ (\sigma\|\mathsf{in}(c,s).\ P\|Q) \rightarrow \nu\overline{n}.\ (\sigma\|P\theta\|Q)} \quad \text{if } \nu\overline{n}.\sigma \vdash s\theta$$

$$\frac{\sigma \models \phi}{\nu\overline{n}.\ (\sigma\|\ \mathsf{if}\ \phi\ \mathsf{then}\ P\ \mathsf{else}\ Q)\ \rightarrow\ \nu\overline{n}.\ (\sigma\ \|\ P)} \qquad \frac{\sigma \not\models \phi}{\nu\overline{n}.\ (\sigma\|\ \mathsf{if}\ \phi\ \mathsf{then}\ P\ \mathsf{else}\ Q)\ \rightarrow\ \nu\overline{n}.\ (\sigma\ \|\ Q)}$$

Example 5 *We develop the transition system from*

$$P = (\nu n_a.\ P_A(a,d,n_a))\|(\nu n_b.\ P_B(a,b,n_b))\|\mathit{out}(c, \mathsf{sk}(d)).\mathbf{0}$$

For convenience, the names n_a, n_b are pushed out and, even if we don't write it explicitly, these names are understood as restricted.

The process P_A may move first, yielding

$$\begin{array}{rl}
P \rightarrow P_1 = \nu n_a, \nu n_b.\ (& \{x_1 \mapsto \{a, n_a\}^p_{\mathsf{pk}(d)}\} \\
\| & (\mathit{in}(c, \{n_a, y\}^p_{\mathsf{pk}(a)}.\mathit{out}(c, \{y\}^p_{\mathsf{pk}(b)}).\mathbf{0} \\
\| & P_B(a,b,n_b) \\
\| & \mathit{out}(c, \mathsf{sk}(d)))
\end{array}$$

The process P_B may also move first, and the resulting general process depends on an input t_1 such that $\emptyset \vdash \{a, t_1\}^p_{\mathsf{pk}(b)}$

$$\begin{array}{rl} P \rightarrow P_2 = \nu n_a, \nu n_b. (& P_A(a, d, n_a) \\ \| & \mathit{out}(c, \{t_1, n_b\}^p_{\mathsf{pk}(a)}.\mathit{in}(c, \{n_b\}^p_{\mathsf{pk}(b)}).\mathbf{0} \\ \| & \mathit{out}(c, \mathsf{sk}(d))) \end{array}$$

The last process may also move first, yielding

$$\begin{array}{rl} P \rightarrow P_3 = \nu n_a, \nu n_b. (& \{x_1 \mapsto \mathsf{sk}(d)\} \\ \| & P_A(a, d, n_a) \\ \| & P_B(a, b, n_b)) \end{array}$$

From the process P_1, there are again three possible transitions:

$$\begin{array}{rl} P_1 \rightarrow P_{11} = \nu n_a, \nu n_b. (& \{x_1 \mapsto \{a, n_a\}^p_{\mathsf{pk}(d)}\} \\ \| & (\mathit{out}(c, \{t_2\}^p_{\mathsf{pk}(b)}).\mathbf{0} \\ \| & P_B(a, b, n_b) \\ \| & \mathit{out}(c, \mathsf{sk}(d))) \end{array}$$

If $\nu n_a, n_b.\{a, n_a\}^p_{\mathsf{pk}(d)} \vdash \{n_a, t_2\}^p_{\mathsf{pk}(a)}$.

$$\begin{array}{rl} P_1 \rightarrow P_{12} = \nu n_a, \nu n_b. (& \{x_1 \mapsto \{a, n_a\}^p_{\mathsf{pk}(d)}\} \\ \| & (\mathit{in}(c, \{n_a, y\}^p_{\mathsf{pk}(a)}.\mathit{out}(c, \{y\}^p_{\mathsf{pk}(b)}).\mathbf{0} \\ \| & \mathit{out}(c, \{t_3, n_b\}^p_{\mathsf{pk}(a)}.\mathit{in}(c, \{n_b\}^p_{\mathsf{pk}(b)}).\mathbf{0} \\ \| & \mathit{out}(c, \mathsf{sk}(d))) \end{array}$$

If $\nu n_a, n_b.\{a, n_a\}^p_{\mathsf{pk}(d)} \vdash \{a, t_3\}^p_{\mathsf{pk}(b)}$.

$$\begin{array}{rl} P_1 \rightarrow P_{13} = \nu n_a, \nu n_b. (& \{x_1 \mapsto \{a, n_a\}^p_{\mathsf{pk}(d)}; x_2 \mapsto \mathsf{sk}(d)\} \\ \| & (\mathit{in}(c, \{n_a, y\}^p_{\mathsf{pk}(a)}.\mathit{out}(c, \{y\}^p_{\mathsf{pk}(b)}).\mathbf{0} \\ \| & P_B(a, b, n_b)) \end{array}$$

We do not continue here the full transition sequence, which is too large to be displayed (there are around 300 nodes in the full tree).

Exercise 5
In the above example, give the complete transition sequence that yields the attack.

In the above example, we see that the transition system might actually be infinite: t_1, t_2, t_3 are arbitrary terms that satisfy some deducibility conditions.

Such deducibility conditions can be simplified (and decided). This the the subject of the next section on bounded process verification.

2.7 Trace security properties

The basic property: secrecy. In any reachable process, what is supposed to remain secret cannot be deduced from the frame.

Example 6 *If we consider the scenarios of example 4,*

- *For the process*

$$P_{01} = (\nu n_a.\ P_A(a,d,n_a))\|(\nu n_b.\ P_B(a,b,n_b))\|\mathsf{out}(c,\mathsf{sk}(d)).\mathbf{0}$$

we typically wish that

$$P_{01} \xrightarrow{*} \nu\overline{n}.(\sigma\|Q) \quad\Longrightarrow\quad \nu\overline{n}.\sigma \not\vdash n_b$$

Since n_b is generated in a process involving two uncorrupted agents. (We cannot however require the secrecy of n_a)

- *Consider now*

$$P_{02} = (\nu n_a.\ \mathsf{in}(c,x_b).\ P_A(a,x_b,n_a))\|(\nu n_b.\ \mathsf{in}(c,x_a).\ P_B(x_a,b,n_b))\|\mathsf{out}(c,\mathsf{sk}(d)).\mathbf{0}$$

Neither n_a nor n_b can be required to remain secret: this depends on the inputs x_a and x_b. Expressing the secrecy requires a honesty predicate or a disequality test. For instance:

$$P_{02} \xrightarrow{*} \nu\overline{n}.(\sigma\|Q) \wedge x_b \neq d \quad\Longrightarrow\quad \nu\overline{n}.\sigma \not\vdash n_a$$

- *For the process*

$$P_{03} = !((\nu n_a.\ \mathsf{in}(c,x_b).\ P_A(a,x_b,n_a))\|(\nu n_b.\ \mathsf{in}(c,x_a).\ P_B(x_a,b,n_b)))\|\mathsf{out}(c,\mathsf{sk}(d)).\mathbf{0}$$

we need extra material in order to express the secrecy: we wish that, in any copy of the process, in which $x_b \neq d$, then n_a is secret. Be careful that x_b is actually a local variable of the thread and should actually be renamed in each copy. Similarly, n_a, n_b are renamed in each instance of the thread.

There are again several ways of specifying the desired properties (we will see later how this is performed in ProVerif): we may split the processes in those for which x_b is bound to a honest party and those in which $x_b = d$ and then forget about the different copies in the specification:

$$\begin{array}{rcl} P'_{03} & = & !((\nu n_a.\ \mathsf{in}(c,x_b).\ \mathsf{if}\ x_b \neq d\ \mathsf{then}\ P_A(a,x_b,n_a))\|(\nu n_b.\ \mathsf{in}(c,x_a).\ P_B(x_a,b,n_b))\| \\ & & \qquad (\nu n'_a.\ \mathsf{in}(c,d).\ P_A(a,d,n'_a))\|(\nu n'_b.\ \mathsf{in}(c,d).\ P_B(d,b,n'_b)))\| \\ & & \mathsf{out}(c,\mathsf{sk}(d)).\mathbf{0} \end{array}$$

n_a, n_b are members of a set $\mathcal{N}_h$ and $n'_a, n'_b \notin \mathcal{N}_h$. Then we require that

$$P'_{03} \xrightarrow{*} \nu\overline{n}.(\sigma\|Q) \quad\Longrightarrow\quad \forall n \in \overline{n} \cap \mathcal{N}_h.\nu\overline{n}.\sigma \not\vdash n$$

Another possibility, which is more general and allows to express more security properties consist in enriching the process algebra in two ways:

  - *Each copy of a thread first gets an identification on a private channel. This id is sent back on a public channel. In this way we can refer to each copy of a thread in a unique way.*

- *The local variables of the threads are recorded in the process calculus. In this way, we may refer to the variable x_b^i, generated in the process whose id is i.*
- *Each thread records in its local memory the names that are local to the process. In this way, we may refer to the name n_a^i, that is generated by the process whose pid is i.*

Then we require for instance:

$$P_{03} \xrightarrow{*} \nu\overline{n}.(\sigma \| Q) \qquad \Longrightarrow \qquad \forall i.\ (x_b^i = d \vee \nu\overline{n}.\sigma \not\vdash n_a^i)$$

We will see later how this is implemented (in a simple way) in ProVerif.

Other trace properties such as agreement properties, refer to relations between different local variables.

Example 7 *Coming back again to the above process P_{03}, we can express that the parties agree on n_a, n_b:*

$$\forall i.\ (x_b^i = b \wedge x^i \neq \perp \Longrightarrow \exists j. x^i = n_b^j \wedge x_a^j = a)$$

In other words, each time x is bound in the ith copy of the thread P_A such that $x_b^i = b$, then there is a copy j of the thread P_B, with $x_a^j = a$, in which the parties share the same value n_b^j.

This property has to hold at any point of time (any reachable state must satisfy the property).

It is known as a non-injective agreement property*: there are stronger agreement properties, that would require the mapping from ids i to ids j to be injective. There are also weaker agreement properties, in which the actual agreement values are not required to be identical, but only their existence is required.*

Such properties are encoded in, e.g., ProVerif by adding events in the processes, that send (on a private channel) the committed value. Somehow, we introduce a referee process who gets the values and compare them.

In the following section, we forget about replication. Then the security properties are much easier to specify, as we will see.

3 Bounded process verification

Assume processes without replication. Then the transition system is finite in depth but might be infinitely branching, as we saw in the example 5. The idea then is to represent in a simple symbolic way the set of terms t_1, t_2, t_3 that satisfy the required conditions.

This is what we formalize now.

3.1 Deducibility constraints

Definition 3.1 *A* Deducibility constraint system *is a conjunction $T_1 \overset{?}{\vdash} t_1 \wedge \ldots \wedge T_n \overset{?}{\vdash} t_n \wedge E$ in which $T_1, \ldots, T_n$ are finite sets of constructor terms, $t_1, \ldots, t_n$ are constructor terms, E is a finite set of equations between constructor terms and such that*

- $T_1 \subseteq T_2 \cdots \subseteq T_n$
- *for every i, $\mathcal{V}(T_i) \subseteq \mathcal{V}(t_1, \ldots, t_{i-1})$*

Intuitively, the sets T_i correspond to messages that have been sent on the network, while $t_1, \ldots, t_n$ are the messages that are expected by the processes, hence have to be constructed by the environment.

The first condition, called *monotonicity* reflects the fact that the set of messages that have been sent on the network can only increase. (In other words, the ordering on the atomic deducibility constraints is a temporal ordering of actions).

The second condition (called *origination*) reflects the properties of our simple processes: variables that occur in a message sent on the network must appear before in messages received from the network.

Thanks to these two properties, for any variable x that occurs in the pure deducibility constraint part C (and not only in the equational part E), there is a unique T_x such that $x \notin \mathcal{V}(T_x)$ and T_x is minimal with respect to inclusion among the sets T such that $T \overset{?}{\vdash} u \in C$ and $x \in \mathcal{V}(u)$.

Such constraint systems may be enriched with equations/disequations between non-constructor terms or other constraints, that correspond to the conditions in the process calculus. We consider (for now) only these simple constraints. If we have time, we will show some extensions.

Definition 3.2 *Given an intruder deduction relation $\vdash$, a substitution σ is a solution of a deducibility constraint system $T_1 \overset{?}{\vdash} t_1 \wedge \ldots \wedge T_n \overset{?}{\vdash} t_n \wedge E$ if, for every i, $T_i\sigma \vdash t_i\sigma$ and, for any equation $s = t$ in E, $s\sigma \downarrow = t\sigma \downarrow$.*

Example 8 *Consider the constraints corresponding to one of the possible NS symbolic trace. For self-containedness, we give explicitly the free names to the attacker and assume that all names that are not explicitly given are (supposedly) secret:*

$$\begin{array}{rcl} a, b, d, \mathsf{sk}(d), \{a, n_a\}^p_{\mathsf{pk}(d)} & \overset{?}{\vdash} & \{a, x\}^p_{\mathsf{pk}(b)} \\ a, b, d, \mathsf{sk}(d), \{a, n_a\}^p_{\mathsf{pk}(d)}, \{x, n_b\}^p_{\mathsf{pk}(a)} & \overset{?}{\vdash} & \{n_a, y\}^p_{\mathsf{pk}(a)} \end{array}$$

The failure of the secrecy of n_b is given by the additional constraint:

$$a, b, d, \mathsf{sk}(d), \{a, n_a\}^p_{\mathsf{pk}(d)}, \{x, n_b\}^p_{\mathsf{pk}(a)}, \{y\}^p_{\mathsf{pk}(d)} \overset{?}{\vdash} n_b$$

Example 9 *Consider the example of the exercise 2 (the trace yielding an attack).*

Let $T_0 = \{\mathsf{sk}(c), a, b, c, \{\left\langle a, \{s\}_{\mathsf{pk}(b)}\right\rangle\}_{\mathsf{pk}(b)}\}$

$$\begin{array}{rcl} T_0 & \overset{?}{\vdash} & \{\left\langle a, \{x_1\}_{\mathsf{pk}(b)}\right\rangle\}_{\mathsf{pk}(b)} \\ T_0, \{\left\langle b, \{x_1\}_{\mathsf{pk}(a)}\right\rangle\}_{\mathsf{pk}(a)} & \overset{?}{\vdash} & \{\left\langle c, \{x_2\}_{\mathsf{pk}(a)}\right\rangle\}_{\mathsf{pk}(a)} \\ T_0, \{\left\langle b, \{x_1\}_{\mathsf{pk}(a)}\right\rangle\}_{\mathsf{pk}(a)}, \{\left\langle a, \{x_2\}_{\mathsf{pk}(c)}\right\rangle\}_{\mathsf{pk}(c)} & \overset{?}{\vdash} & \{\left\langle c, \{x_3\}_{\mathsf{pk}(a)}\right\rangle\}_{\mathsf{pk}(a)} \\ T_0, \{\left\langle b, \{x_1\}_{\mathsf{pk}(a)}\right\rangle\}_{\mathsf{pk}(a)}, \{\left\langle a, \{x_2\}_{\mathsf{pk}(c)}\right\rangle\}_{\mathsf{pk}(c)}, \{\left\langle a, \{x_3\}_{\mathsf{pk}(c)}\right\rangle\}_{\mathsf{pk}(c)} & \overset{?}{\vdash} & s \end{array}$$

Exercise 6
What are the solutions of

$$T_0 \stackrel{?}{\vdash} \{\left\langle a, \{x_1\}_{\mathsf{pk}(b)} \right\rangle\}_{\mathsf{pk}(b)}$$

Example 10 *We may specify the violation of agreement properties. For instance in the NS protocol, we consider two instances of P_A: $P_A(a, b, n_a)$, $P_A(a, d, n'a)$ and $P_B(a, b, n_b)$.*

$$\begin{array}{rcl} \{a, n'_a\}^p_{\mathsf{pk}(d)}, a, b, d, \mathsf{sk}(d) & \stackrel{?}{\vdash} & \{a, x\}^p_{\mathsf{pk}(b)} \\ \{a, n'_a\}^p_{\mathsf{pk}(d)}, a, b, d, \mathsf{sk}(d), \{x, n_b\}^p_{\mathsf{pk}(a)} & \stackrel{?}{\vdash} & \{n'_a, y\}^p_{\mathsf{pk}(a)} \\ \{a, n'_a\}^p_{\mathsf{pk}(d)}, a, b, d, \mathsf{sk}(d), \{x, n_b\}^p_{\mathsf{pk}(a)}, \{y\}^p_{\mathsf{pk}(b)} & \stackrel{?}{\vdash} & \{n_b\}^p_{\mathsf{pk}(b)} \end{array}$$

The agreement (on n_a) is violated if the above constraint system, together with $x \neq n_a$ is satisfiable. Indeed, in such a case, b believes that he agreed with a on the value x, while a believes that she agreed with b on n_a, and the two values are distinct.