Revisiting trace and may testing equivalences

In this exercise, we consider a fragment of the applied pi-calculus of the lecture notes, given by the following grammar:

\[ P, P' ::= 0 \mid \text{in}(c, x).P \mid \text{out}(c, u).P \mid \text{let } x = t \text{ in } P \text{ else } P' \]

In other words, we do not consider name creation, replication, and even parallel composition. As before, \( u \) denotes a constructor term, while \( t \) may feature destructors.

**Semantics** The definition of internal reduction is unchanged, except for the fact that several cases become useless. The labelled transition system has the standard rules for input, output, let, plus the following new rule:

\[ (P, \Phi) \xrightarrow{\text{store}(w, t)} (P, \Phi \uplus \{w \mapsto \text{u}\}) \quad \text{if} \quad w \in \mathcal{W} \setminus \text{dom}(\Phi), t \in \mathcal{T}(\text{dom}(\Phi) \cup \mathcal{N} \setminus \text{bn}(\Phi)) \quad \text{and} \quad t \Phi \Downarrow \text{u} \]

Since a process cannot create new private names, the bound names of a frame (i.e. \( \text{bn}(\Phi) \)) is never modified. We will thus assume in the rest of this exercise that all frames have the same bound names. This amounts to have a static distinction between public and private names, where only public names may appear in recipes: we thus have a set of public names \( \mathcal{N}_{\text{pub}} \) such that, for all \( \Phi \), \( \mathcal{N}_{\text{pub}} = \mathcal{N} \setminus \text{bn}(\Phi) \). A recipe is simply a term of \( \mathcal{T}(\mathcal{W} \cup \mathcal{N}_{\text{pub}}) \).

The only transitions labelled \( \tau \) correspond to the evaluation of let constructs. The label \( \text{store}(w, t) \) is considered observable.

**Trace equivalence** We define as before \( A \xrightarrow{\text{tr}} B \) when \( \text{tr} \) contains no \( \tau \) action and \( A \xrightarrow{\text{tr}'} B \) for some \( \text{tr}' \) obtained from \( \text{tr} \) by inserting \( \tau \) actions. We finally define trace equivalence as before: \( A \approx B \) iff \( \text{Tr}'(A) = \text{Tr}'(B) \) where

\[ \text{Tr}'(P, \Phi) \overset{\text{def}}{=} \{(\text{tr}, \Phi') : (P, \Phi) \xrightarrow{\text{tr}, \Phi} (P', \Phi') \text{ and } \Phi' \sim \Phi'\} \]

**May testing** We consider a simplified notion of may testing, relying on a restricted syntax for tests\(^1\). We first define a grammar for test contexts \( C \) (with a hole noted \( [] \)) and terminal tests \( D \) (where \( \_ \) stands for a dummy variable that cannot be used elsewhere):

\[
C ::= \text{in}(c, w).C \mid \text{out}(c, w).C \mid \text{let } w = t \text{ in } C \text{ else } 0 \mid [] \\
D ::= \text{out}(T, w) \mid \text{let } _\_ = t \text{ in } D \text{ else } 0 \mid \text{let } _\_ = t \text{ in } 0 \text{ else } D
\]

As before, \( T \) is the special channel used to signal success. We then define tests as those processes that can be written \( C[D] \) for some test context \( C \) and terminal test \( D \), that do not feature private names, but may feature free variables in \( W \). Note that tests only use variables in \( W \), and that

---

\(^1\)This is actually without loss of generality, but we won’t show it here.
outputs in tests are restricted to variables. We may also assume that a test never features two bindings for the same variable \( w \).

We say that a process \( Q \) succeeds when \( Q \xrightarrow{\ast} \text{out}(T, u) \) for some \( u \), and we define:

\[
T(P, \Phi) = \{ T : T \text{ is a test with } \text{fv}(T) \subseteq \text{dom}(\Phi) \text{ and } (T\Phi | P) \text{ succeeds} \}
\]

Finally, \( A \) and \( B \) are may testing equivalent when \( T(A) = T(B) \).

**Question 1.** We say that a frame \( \Phi \) satisfies a terminal test \( D \), noted \( \Phi \models D \), when \( D \Phi \) succeeds. Show that, for all frames \( \Phi \) and \( \Psi \) such that \( \Phi \sim \Psi \), we have \( \Phi \models D \) iff \( \Psi \models D \).

**Question 2.** Define a translation \( \text{tr}(. ) \) from test contexts to traces such that, for all test context \( C \) and terminal test \( D \), we have \( C[D] \in T(A) \) iff there exists \( A' \) such that \( A' \xrightarrow{\Phi} A' \) and \( \Phi(A') \models D \). Justify the two directions of this equivalence.

**Question 3.** Conclude that trace equivalence implies may testing equivalence\(^2\).

For the rest of the exercise, we assume that we have a destructor implementing equality checking (i.e. \( \text{eq}(x, x) \rightarrow x \)) so that \( \Phi \sim \Psi \) is equivalent to the following condition alone:

\[
\forall R \in T(W \cup N_{\text{pub}}), R \Phi \downarrow \text{ iff } R \Psi \downarrow
\]

**Question 4.** Let \( \Phi \) be a frame and \( (\Psi_i)_{i \in [1:n]} \) be a finite collection of frames such that, for all \( i \in [1:n] \), \( \Phi \not\sim \Psi_i \). Show that there exists \( D \) such that \( \Phi \models D \) but, for all \( i \in [1:n] \), \( \Psi_i \not\models D \).

**Question 5.** A configuration \( A \) is image finite if, for all observable traces \( tr \), the set \( \{ \Phi(B) : A \xrightarrow{\Phi} B \} \) is finite up to static equivalence. Show that, for all image finite configurations \( A \) and \( B \) such that \( T(A) = T(B) \), one has \( A \approx B \). *Hint: proceed by contradiction.*

## 2 Bana-Comon logic: EUF and LAK

Consider a keyed hash function \( H(\_ , \_) \). We say that it satisfies existential unforgeability (EUF) when, for any attacker \( A \), the probability \( \text{Pr}[\text{EUF}(A) = 1] \) is negligible in \( \eta \), where the experiment EUF is defined as follows:

\[
\begin{align*}
\text{EUF}(A) & \overset{\text{def}}{=} k \overset{\$}{\leftarrow} \{0, 1\}^n \\
Q & \leftarrow \emptyset \\
\text{if } m \not\in Q \text{ and } s = H(m, k) \text{ then return } 1 \\
\text{return } H(x, k)
\end{align*}
\]

Here, \( O \) is an oracle allowing the attacker to compute hashes using the (private) key \( k \) that is randomly sampled at the beginning of the experiment. The experiment succeeds if the attacker manages to compute the hash \( s \) of a message \( m \) without having called \( O(m) \).

We consider the language of the Bana-Comon logic. We write \( t \equiv t' \) for \( \text{EQ}(t, t') \). We write \( t \land t' \) for \( (\text{if } t \text{ then (if } t' \text{ then true else false) else false} \), and similarly for \( t \lor t' \) and \( \neg t \).

**Question 1.** Show that the following rule is unsound:

\[
\begin{align*}
t[b] & \sim t' \quad a \sim b \\
\hline
\not{t[a]} & \sim t'
\end{align*}
\]

\(^2\)In the lecture notes, we have shown that this is not the case if non-deterministic computations are allowed: this problem disappears here, not due to abusive simplifications but mainly due to the store actions.
Question 2. Let $h$ be a binary function symbol and $k$ a name, which will be used as a key for $h$. Let $m$ and $s$ be two terms in which $k$ only appears as second argument of $h$. Let $E = \{ m' : h(m', k) \text{ is a subterm of } m \text{ or } s \}$. Consider the following inference rule$^3$:

$$\text{if } \forall m' \in E \; m \triangleq m' \text{ then false else } s \triangleq h(m, k) \sim \text{false}$$

Show that the conclusion formula is satisfied in all computational interpretations where the interpretation of $h$ satisfies EUF. In other words, the rule is sound wrt. this class of models.

In order to derive equivalences, we will consider the previous rule, together with the following ones, which are also sound:

$$t \sim t \quad t[b] \sim t' \quad a \equiv b \sim \text{true} \quad \text{if } b \text{ then } t' \text{ else } c \sim \text{true} \quad \text{if } t' \text{ then } t \text{ else true } \sim \text{true}$$

$$t \sim t' \quad \neg t \sim \neg t' \quad t \triangleq t' \sim \text{true} \quad \pi_1((t, t')) = t \sim \text{true} \quad \pi_2((t, t')) = t' \sim \text{true} \quad u \equiv v$$

In the last rule $u \equiv v$ means that $u$ and $v$ are logically equivalent expressions built from (boolean) variables and conditionals if ... then ... else ... For example, that rule allows to derive all statements of the form $(\text{if } b \text{ then } c \text{ else } d) \equiv (\text{if not}(b) \text{ then } d \text{ else not(not}(c))) \sim \text{true}$.

Finally, equivalences $t \sim t'$ and equalities $t \equiv t'$ may be considered implicitly modulo symmetry.

Consider the following frames, resulting from an execution of the LAK protocol:

$$\Phi_1 \stackrel{\text{def}}{=} nR$$
$$\Phi_2 \stackrel{\text{def}}{=} \Phi_1, (n_T, h((g_1(\Phi_1), n_T), k))$$

We then consider the following boolean terms:

$$t_a \stackrel{\text{def}}{=} \pi_2(g_2(\Phi_2)) \equiv h((n_R, \pi_1(g_2(\Phi_2))), k)$$
$$t_h \stackrel{\text{def}}{=} g_1(\Phi_1) \equiv n_R \land \pi_1(g_2(\Phi_2)) \equiv n_T$$
$$t_c \stackrel{\text{def}}{=} (g_1(\Phi_1), n_T) \equiv (n_R, \pi_1(g_2(\Phi_2)))$$

The first term corresponds to the acceptance test that the reader would evaluate when receiving $g_2(\Phi_2)$ after having sent $n_R$. The second term expresses that the interaction is honest enough, and more precisely that the tag and reader agree on their respective nonces.

Question 3. Derive if $t_a$ then $t_c$ else true $\sim$ true using the above rules.

Question 4. Conclude that there exists a derivation of the following equivalence, which intuitively expresses that if the reader accepts an interaction, then the interaction is honest:

$$\text{if } t_a \text{ then } t_h \text{ else true } \sim \text{true}$$

Question 5. We modify our frames to reason about an execution involving more reader messages, with a subterm $m$ that will be defined below:

$$\Phi_1 \stackrel{\text{def}}{=} nR, nR'$$
$$\Phi_2 \stackrel{\text{def}}{=} \Phi_1, (n_T, h((g_1(\Phi_1), n_T), k))$$
$$\Phi_3 \stackrel{\text{def}}{=} \Phi_2, h((m, n_R), k)$$

$^3$If $E$ is empty, the disjunction $\forall m' \in E \; m = m'$ is simply false.
(a) We want $\Phi_3$ to model the situation where the first reader has accepted its input, and responded with its second output. Give the term $m$ corresponding to that situation, relying on $g_2(\Phi_2)$ to model the reader’s input.

We define $t_a$ as $f_a$ above, except for the modification of $\Phi_1$ which now features two nonces. We similarly adapt $t_b$ into $t_b^1$. It is easy to see that we still have if $t_a^1$ then $t_b^1$ else true $\sim$ true.

(b) Now the second reader may receive a message $g_3(\Phi_3)$. Give the boolean term $t_a^2$ corresponding to the acceptance condition of the second reader.

(c) Let $t_a^2 \defeq g_1(\Phi_1) = n_R' \land \pi_1(g_3(\Phi_3)) = n_T$. Does if $t_a^1 \land t_a^2$ then $t_b^2$ else true $\sim$ true hold? No formal derivation is required, but an argument using the EUF assumption should be provided.

3 Constraint solving and resolution for a naive protocol

Consider the following protocol, where $a$ and $b$ are respective secrets of agents $A$ and $B$, keys $pk(a)$ and $pk(b)$ are public, and $n$ is a nonce generated by $A$:

$$
A \rightarrow B : \langle pk(a), aenc(n, pk(b)) \rangle \\
B \rightarrow A : \langle pk(b), aenc(n, pk(a)) \rangle
$$

The naive idea is that $A$ sends a first message containing a component that identifies the sender, and one that transmits $n$ securely to the intended recipient $B$. Upon receipt of a message of the form $(x, aenc(y, pk(b)))$, $B$ obtains $n$ and sends back the last message to $x$ as an acknowledgment. Obviously, there is an attack: an outsider can learn $n$.

**Question 1.** We want to model this protocol using the applied pi-calculus. We model pairs and asymmetric encryption using destructors. Give two processes $A(\text{sk}_a, pk_b)$ and $B(\text{sk}_b)$ modelling the two roles of the protocol, where $A$ only uses channel $c_A$ and $B$ only uses $c_B$. Process $A$ should have $n$ as a free name.

**Question 2.** Let $P \defeq (A(a, pk(b)) \mid B(b))$ and $\Phi \defeq (n, a, b).\{w_a \mapsto pk(a), w_b \mapsto pk(b)\}$. Give the symbolic configuration corresponding to the execution of each of the following traces, starting with the configuration $(P, \Phi, \emptyset)$:

$$
\text{tr}_1 \defeq \text{out}(c_A, w).\text{in}(c_B, x) \quad \text{tr}_2 \defeq \text{tr}_1.\text{out}(c_B, w')
$$

You should make the most general choices (e.g. in the symbolic evaluation steps) so that the resulting symbolic configuration accounts for all concrete configurations, in the sense of the completeness result of the symbolic semantics.

**Question 3.** Let $(\_ \Psi, C)$ be the last configuration obtained above. Show that the constraint system $C \land \Psi \vdash? n$ can be simplified into a solved constraint system. Conclude that $n$ is not secret.

**Question 4.** Give the clauses that Proverif would generate for $P$ and the theory of pairs and asymmetric encryption.

**Question 5.** We consider resolution with selection, with the selection strategy where any hypothesis that is not of the form $\mathsf{att}(x)$ is selected. Show how this rule can be used to derive, from the previous clauses, a solved clause with conclusion $\mathsf{att}(n)$. The hypotheses of a clause may be treated as a set of atoms, e.g. multiple hypotheses can be merged.