Addendum to “Service Automata”

– formal proofs –

Richard Gay, Heiko Mantel, Barbara Sprick

Department of Computer Science, TU Darmstadt, Germany

{gay,mantel,sprick}@mais.informatik.tu-darmstadt.de

November 16, 2011

This document contains the formal proof for Theorem 1 in the article “Service Automata” [1] (Section 2) and preliminaries for the formalism (Section 1).

1 Preliminaries

In the proof, we denote the length of a sequence s by $|s|$, and the projection of a sequence s to an alphabet E of events by $s \restriction E$. As a shorthand, we write $e_1, e_2 \lhd t$ for $e_1 \lhd t$ and $e_2 \lhd t$.

1.1 A Primer to Hoare’s Communicating Sequential Processes

We briefly recall the sublanguage of Hoare’s Communicating Sequential Processes (CSP) used in this article. For a proper introduction, we refer to [2].

A *process* P is a pair $(\alpha(P), \mathit{traces}(P))$ consisting of a set of events and a nonempty, prefix-closed set of finite sequences over $\alpha(P)$. The *alphabet* $\alpha(P)$ contains all events in which P could in principle engage. The *set of possible traces* $\mathit{traces}(P) \subseteq (\alpha(P))^*$ contains all sequences of events that the process could in principle perform. We use $\langle\,\rangle$ to denote the empty sequence, $\langle e\rangle$ to denote the trace consisting of the single event e, and $s.t$ to denote the concatenation of two traces s and t. That an event e occurs in a trace t is denoted by $e \lhd t$.

The CSP process expression STOP_E specifies a process with alphabet E and a set of traces containing only $\langle\,\rangle$. A process that performs event e and then behaves according to the process expression P, is specified by $e \rightarrow P$. External and internal choice between P and Q are specified by $P \,\square\, Q$ and $P \sqcap Q$, respectively. They model that the process behaves according to either P or Q. The parallel composition of P and Q is specified by $P \parallel Q$. The parallel processes have to synchronize on the occurrences of all events that their alphabets have in common. The process $P \setminus E$ behaves as P but all events in the set E are hidden by removing them from the process’ alphabet and possible traces.

The binary operators $\square$, $\sqcap$ and $\parallel$ are lifted to n-ary operators over non-empty finite index sets. For instance, $\square_{x\in X}\, P(x)$ equals $P(a)$ if $X = \{a\}$ and equals $P(a) \,\square (\square_{x\in X\setminus\{a\}}\, P(x))$ if $a \in X$ and X contains at least two elements.

We use structured events of the form $\mathsf{c}.m$ to model the communication of a message m on a channel c. In a process expression we write $\mathsf{c}!m$ instead of $\mathsf{c}.m$ in order to indicate that message m is sent on c, and use $\mathsf{c}?x\colon M$ for receiving some message $m \in M$ on channel c while instantiating the variable x with m. Effectively, $\mathsf{c}?x\colon M \to P(x)$ corresponds to an external choice on the events in $\{c.m \mid m \in M\}$ such that the computation continues according to $P(m)$.

A process definition $\mathsf{NAME} \overset{\text{def}}{=}_\alpha P$ declares a new process name NAME and defines that NAME models a process whose traces are given by the process expression P and whose alphabet equals α. We omit the subscript α in a process definition if the alphabet of NAME shall equal $\alpha(P)$. Process names can be used as subexpressions within process expressions, thus allowing for recursion.

Properties of CSP processes are modeled by unary predicates on traces. We say that a unary predicate φ on traces is *satisfied* by a process P (denoted by $P\,\mathsf{sat}\,\varphi$) if and only if $\varphi(t)$ holds for each $t \in \mathit{traces}(P)$.

In the proofs, we denote by $P \,/\, t$ the process P after engaging in the sequence of events t.

1.2 Semantics of CSP

The following lists the trace semantics for the CSP constructs used in the paper.

- for every set E of events, $\alpha(\mathsf{STOP}_E) = E$ and $\mathit{traces}(\mathsf{STOP}_E) = \{\langle\rangle\}$;
- for process P and event $e \in \alpha(P)$, $\alpha(e \to P) = \alpha(P)$ and $\mathit{traces}(e \to P) = \{\langle\rangle\} \cup \{\langle e\rangle.tr \mid tr \in \mathit{traces}(P)\}$;
- for process P, channel c, and message m, if $\mathsf{c}.m \in \alpha(P)$, then $\alpha(\mathsf{c}!m \to P) = \alpha(P)$ and $\mathit{traces}(\mathsf{c}!m \to P) = \{\langle\rangle\} \cup \{\langle \mathsf{c}.m\rangle.tr \mid tr \in \mathit{traces}(P)\}$;
- for channel c, set M, and process expression $P(x)$ parametric in $x \in M$, if $\alpha(P(m)) = \alpha(P(m'))$ and $\mathsf{c}.m \in \alpha(P(m))$ holds for all $m, m' \in M$, then $\alpha(\mathsf{c}?x\colon M \to P(x)) = \alpha(P(m))$ for any $m \in M$ and $\mathit{traces}(\mathsf{c}?x\colon M \to P(x)) = \{\langle\rangle\} \cup \{\langle \mathsf{c}.x\rangle.tr \mid x \in M \wedge tr \in \mathit{traces}(P(x))\}$;
- for processes P and Q with $\alpha(P) = \alpha(Q)$, $\alpha(P \,\Box\, Q) = \alpha(P) = \alpha(Q)$ and $\mathit{traces}(P \,\Box\, Q) = \mathit{traces}(P) \cup \mathit{traces}(Q)$;
- for a non-empty finite set X and a process expression $P(x)$ parametric in x, if $\alpha(P(x)) = \alpha(P(y))$ for all $x, y \in X$, then $\alpha(\Box_{x\in X}\, P(x)) = \alpha(P(y))$ for any $y \in X$ and $\mathit{traces}(\Box_{x\in X}\, P(x)) = \bigcup_{x\in X} \mathit{traces}(P(x))$;
- for processes P and Q, $\alpha(P \parallel Q) = \alpha(P) \cup \alpha(Q)$ and $\mathit{traces}(P \parallel Q) = \{t \in \alpha(P \parallel Q)^* \mid t \upharpoonright \alpha(P) \in \mathit{traces}(P) \,\wedge\, t \upharpoonright \alpha(Q) \in \mathit{traces}(Q)\}$;
- for process P and set E of events, $\alpha(P \setminus E) = \alpha(P) \setminus E$ and $\mathit{traces}(P \setminus E) = \{t \upharpoonright \alpha(P \setminus E) \mid t \in \mathit{traces}(P)\}$.

In the trace semantics, we define two processes P and Q to be equal (written $P = Q$), if and only if $\alpha(P) = \alpha(Q)$ and $\mathit{traces}(P) = \mathit{traces}(Q)$ holds. The following recapitulates some basic properties of CSP processes provided by Hoare in [2].

Lemma 1. *Let P, Q be CSP processes, E, E' be sets of events, tr be a sequence of events, $e \in \alpha(P)$ be an event, and f be an injective alphabet renaming function. Then*

(a) $f(P \parallel Q) = f(P) \parallel f(Q)$ *[2, page 65 (L3)]*
(b) $P \setminus (E \cup E') = (P \setminus E) \setminus E'$ *[2, page 92 (L2)]*
(c) $(P \parallel Q) \setminus E = (P \setminus E) \parallel (Q \setminus E)$ *if* $\alpha(P) \cap \alpha(Q) \cap E = \emptyset$ *[2, page 92 (L6)]*
(d) $f(P \setminus E) = f(P) \setminus f(E)$ *[2, page 92 (L7)]*
(e) $P \,/\, (s_1.s_2) = (P \,/\, s_1) \,/\, s_2$ *[2, page 32 (L2)]*
(f) $(e \to P) \,/\, \langle e\rangle.tr = P \,/\, tr$ *[2, page 32 (L2+L3)]*
(g) $(P \,\Box\, Q) \,/\, tr = P \,/\, tr$ *if* $tr \in \mathit{traces}(P) \setminus \mathit{traces}(Q)$ *and*
$(P \,\Box\, Q) \,/\, tr = Q \,/\, tr$ *if* $tr \in \mathit{traces}(Q) \setminus \mathit{traces}(P)$ *[2, page 88 (L2)]*

2 Proof of Theorem 1

In this section, we give a formal proof for [1, Theorem 1], which shows that the service automata framework SYSTEM soundly enforces the Chinese Wall security policy.

Theorem 1. SYSTEM sat *ChW*.

The proof for the theorem can be found at the end of this section on page 9. To increase the readability of the proof, we show major steps of the proof in separate lemmas:

- Lemma 2 states that the hiding in the definitions of the service automata and the local policies can be moved outside of the controlled system, if local channels are renamed to be unique. Definition 1 specifies the renaming and the controlled system without hiding. This change of the hiding simplifies the reasoning about the events that are hidden in SYSTEM.
- Lemma 3 states that certain events cannot occur in traces of the instantiated service automata: (a) decisions are not sent on channel ddec, (b) requests are not sent on channel rdec, (c) requests and decisions are not forwarded, (d) remote decisions are not approved for locally decidable events, and (e) local enforcement decisions are not sent to an enforcer for events that have a remote responsible node.
- Lemma 4 states that an enforcer performs a critical event only after this enforcer has received a corresponding decision.
- Lemma 5 states that every performed critical event must have ultimately been permitted by its responsible node.
- Lemma 6 states that no decision-making component permits two conflicting events in a single trace.

In the remainder of this section, we abbreviate $\mathsf{COR}_i(CE_i, ED_i)$ by COR_i, $\mathsf{INT}_i(\alpha(\mathsf{PROV}_i), CE_i)$ by INT_i, and $\mathsf{REPLACE}(CE_i)$ by $\mathsf{REPLACE}_i$. By DUM, we denote the set $\{dummy_{ev} \mid ev \in CE\}$ of all dummy events.

Definition 1. *(a) Let $i \in Id$ be an identifier. We define the set of events hidden by the service automaton at node i as*

$$H_i := \left\{ \begin{array}{l|l} \mathsf{sync}.\checkmark, \mathsf{icpt}.ev, \mathsf{enf}.ed, \mathsf{lreq}.ev, \mathsf{rreq}.dr, \mathsf{fwd}.(k, dr), & ev \in CE_i, ed \in ED, \\ \mathsf{edec}.ed, \mathsf{appv}.ed, \mathsf{ddec}.(k, dr), \mathsf{rdec}.(k, dr) & dr \in DR, k \in Id \setminus \{i\} \end{array} \right\}.$$

(b) For all identifiers $i \in Id$, let $\rho_i : E_i \to E_i \cup \{\mathsf{c}_i.m \mid \mathsf{c}.m \in H_i \cup H_i^{\mathrm{pol}}\}$ be a renaming function *on $E_i := \alpha((\mathsf{PROV}_i \parallel \mathsf{INT}_i) \setminus CE_i) \cup \alpha(\mathsf{REPLACE}_i) \cup \alpha(\mathsf{COR}_i) \cup \alpha(\mathsf{DEL}_i) \cup \alpha(\mathsf{DEC}_i(\emptyset)) \cup \alpha(\mathsf{SRP}_i)$ such that*

$$\rho_i(e) = \begin{cases} \mathsf{c}_i.m & \text{if } e \in H_i \cup H_i^{\mathrm{pol}} \text{ with } e = \mathsf{c}.m, \\ e & \text{otherwise.} \end{cases}$$

Note that H_i^{pol} is defined in [1, page 12]. We lift the renaming functions from events to processes as in [2, Section 2.6], by applying the renaming to all events occurring in the respective process.

(c) The set of all renamed hidden events *is defined as $HS := \bigcup_{i \in Id} \rho_i(H_i \cup H_i^{\mathrm{pol}})$,*

(d) The system with internal events*, SI, is defined as*

$$\mathsf{SI} \stackrel{\text{def}}{=} \mathop{\parallel}_{i \in Id} \left(\begin{array}{c} \rho_i\left((\mathsf{PROV}_i \parallel \mathsf{INT}_i) \setminus CE_i\right) \parallel \rho_i(\mathsf{REPLACE}_i) \\ \parallel \rho_i(\mathsf{COR}_i) \parallel \ \rho_i(\mathsf{DEL}_i) \parallel \rho_i(\mathsf{DEC}_i(\emptyset)) \parallel \rho_i(\mathsf{SRP}_i) \end{array} \right).$$

Lemma 2. *If $\alpha(\mathsf{PROV}_i) \cap (HS \cup H_i^{\mathrm{pol}}) = \emptyset$ holds for all identifiers $i \in Id$, then $\mathsf{SI} \setminus HS = \mathsf{SYSTEM}$.*

Proof. For all identifiers $j \in Id$, the definitions of all processes $P \in \{\mathsf{COR}_i, \mathsf{DEL}_i, \mathsf{DEC}_i, \mathsf{SRP}_i \mid i \in Id \setminus \{j\}\}$ ensure $\alpha(P) \cap \rho_j(H_j \cup H_j^{\mathrm{pol}}) = \emptyset$. The precondition $\alpha(\mathsf{PROV}_i) \cap (HS \cup H_i^{\mathrm{pol}}) = \emptyset$ together with $CE_i \subseteq \alpha(\mathsf{PROV}_i)$ [1, page 11] and the definitions of the processes $\mathsf{REPLACE}_i$ and INT_i implies $\alpha(Q) \cap \rho_j(H_j \cup H_j^{\mathrm{pol}}) = \emptyset$ for all $Q \in \{\mathsf{REPLACE}_i, (\mathsf{PROV}_i \parallel \mathsf{INT}_i) \setminus CE_i \mid i \in Id \setminus \{j\}\}$.

We denote the above two observations as (†) below.

$$
\begin{aligned}
\mathsf{SI} \setminus HS = {} & \text{by Definition 1 (d) and Lemma 1 (a)} \\
& \left(\mathop{\|}_{i \in Id} \rho_i \left(\begin{array}{l} ((\mathsf{PROV}_i \,\|\, \mathsf{INT}_i) \setminus CE_i) \,\|\, \mathsf{REPLACE}_i \\ \|\ \mathsf{COR}_i \,\|\ \ \mathsf{DEL}_i \,\|\, \mathsf{DEC}_i(\emptyset) \,\|\, \mathsf{SRP}_i \end{array} \right) \right) \setminus HS \\
= {} & \text{by Definition 1 (c), Lemma 1 (c), and observations (†)} \\
& \mathop{\|}_{i \in Id} \left(\rho_i \left(\begin{array}{l} ((\mathsf{PROV}_i \,\|\, \mathsf{INT}_i) \setminus CE_i) \,\|\, \mathsf{REPLACE}_i \\ \|\ \mathsf{COR}_i \,\|\ (\mathsf{DEL}_i \,\|\, \mathsf{DEC}_i(\emptyset) \,\|\, \mathsf{SRP}_i) \end{array} \right) \setminus \rho_i(H_i \cup H_i^{\mathrm{pol}}) \right) \\
= {} & \text{by Lemma 1 (d)} \\
& \mathop{\|}_{i \in Id} \rho_i \left(\left(\begin{array}{l} ((\mathsf{PROV}_i \,\|\, \mathsf{INT}_i) \setminus CE_i) \,\|\, \mathsf{REPLACE}_i \\ \|\ \mathsf{COR}_i \,\|\ (\mathsf{DEL}_i \,\|\, \mathsf{DEC}_i(\emptyset) \,\|\, \mathsf{SRP}_i) \end{array} \right) \setminus (H_i \cup H_i^{\mathrm{pol}}) \right) \\
= {} & \text{by Definition 1 (b), which gives } \rho_i(e) = e \text{ for all } e \notin H_i \cup H_i^{\mathrm{pol}} \\
& \mathop{\|}_{i \in Id} \left(\begin{array}{l} ((\mathsf{PROV}_i \,\|\, \mathsf{INT}_i) \setminus CE_i) \,\|\, \mathsf{REPLACE}_i \\ \|\ \mathsf{COR}_i \,\|\ (\mathsf{DEL}_i \,\|\, \mathsf{DEC}_i(\emptyset) \,\|\, \mathsf{SRP}_i) \end{array} \right) \setminus (H_i \cup H_i^{\mathrm{pol}}) \\
= {} & \text{by Lemma 1 (b); by precondition } \alpha(\mathsf{PROV}_i) \cap H_i^{\mathrm{pol}} = \emptyset; \\
& \text{by the fact that for all } i \in Id \text{ and } P \in \{\mathsf{INT}_i, \mathsf{REPLACE}_i, \mathsf{COR}_i\},\ \alpha(P) \cap H_i^{\mathrm{pol}} = \emptyset \\
& \text{holds according to the definition of } P\text{; and by definition of } \mathsf{POL}_i \text{ [1, page 12]} \\
& \mathop{\|}_{i \in Id} \big(((\mathsf{PROV}_i \,\|\, \mathsf{INT}_i) \setminus CE_i) \,\|\, \mathsf{POL}_i \,\|\, \mathsf{COR}_i \,\|\, \mathsf{REPLACE}_i \big) \setminus H_i \\
= {} & \text{by definition of } \mathsf{SA}_i \text{ [1, page 6] and } \mathsf{SYSTEM} \text{ [1, page 13]} \\
& \mathop{\|}_{i \in Id} \mathsf{SA}_i(\mathsf{PROV}_i, CE_i, \mathsf{POL}_i, ED_i, \mathsf{REPLACE}_i) = \mathsf{SYSTEM} \qquad \square
\end{aligned}
$$

Remark 1. *In the following, we repeatedly make use of the following patterns of reasoning.*

- *Let $tr \in traces(\mathsf{SI})$ be a trace, P, Q be processes such that $\mathsf{SI} = P \,\|\, Q$ holds, and $e \in \alpha(Q)$ be an event. Then from $e \lhd tr$ we can follow that $e \lhd tr \upharpoonright \alpha(Q) \wedge tr \upharpoonright \alpha(Q) \in traces(Q)$. We indicate this reasoning by $\stackrel{(*)}{\Longrightarrow}$.*
- *Let Q be a process, $e \in \alpha(Q)$ be an event, tr_Q be a trace, and $E \subseteq \alpha(Q)$ be the set of immediate predecessors of e in process Q. Then from $e \lhd tr_Q \wedge tr_Q \in traces(Q)$ we can conclude that $\bigvee_{e' \in E}(e' \lhd tr_Q)$ holds. We use $\stackrel{(**)}{\Longrightarrow}$ to indicate this reasoning.*
- *We often combine the above steps in the form $e \lhd tr \stackrel{(*)}{\Longrightarrow} e \lhd tr_Q \wedge tr_Q \in traces(Q) \stackrel{(**)}{\Longrightarrow} \bigvee_{e' \in E}(e' \lhd tr_Q) \implies \bigvee_{e' \in E}(e' \lhd tr)$ for $tr_Q = tr \upharpoonright \alpha(Q)$. The last implication trivially holds. In the remainder of this section, we abbreviate such chains of reasoning by $e \lhd tr \stackrel{Q}{\Rightarrow} \bigvee_{e' \in E}(e' \lhd tr)$.*

Lemma 3. *Let $CE_i^{ld} := \{ev \in CE_i \mid id(ev) = resp(ev)\}$ be the set of locally decidable critical events at node $i \in Id$. Then for all traces $tr \in \mathsf{SI}$ and identifiers $i, j, k \in Id$ with $j \neq i$, the following holds:*

(a) For all decisions $ed \in ED$, it holds that $\mathsf{ddec}_i.(j, (k, ed)) \ntriangleleft tr$.
(b) For all events $ev \in CE$, it holds that $\mathsf{rdec}_i.(j, (k, ev)) \ntriangleleft tr$.
(c) For all events and decisions $x \in CE \cup ED$, it holds that $\mathsf{fwd}_j.(i, (i, x)) \ntriangleleft tr$.
(d) For all decisions $ed \in CE_i^{ld} \times CE_i$, it holds that $\mathsf{appv}_i.ed \ntriangleleft tr$.
(e) For all decisions $ed \in ED_i \setminus (CE_i^{ld} \times CE_i)$ it holds that $\mathsf{edec}_i.ed \ntriangleleft tr$.

Proof. Let the trace $tr \in traces(\mathsf{SI})$ be arbitrary but fixed.

(a) Let identifiers $i, j, k \in Id$ with $i \neq j$ and decision $ed \in ED$ be arbitrary but fixed. We show the claim by contradiction and assume $\mathsf{ddec}_i.(j, (k, ed)) \lhd tr$.

$$\begin{aligned}
&\xRightarrow{(*)} \mathsf{ddec}_i.(j,(k,ed)) \lhd tr \upharpoonright \alpha(\rho_i(\mathsf{SRP}_i)) \wedge tr \upharpoonright \alpha(\rho_i(\mathsf{SRP}_i)) \in traces(\rho_i(\mathsf{SRP}_i))\\
&\Longrightarrow \text{by definition of } \mathsf{SRP}_i\\
&\quad (k, ed) \in Id \times CE\\
&\Longrightarrow \text{by the precondition } Id \times CE \cap Id \times ED = \emptyset \text{ for } \mathsf{SRP} \text{ [1, page 9]}\\
&\quad (k, ed) \notin Id \times ED
\end{aligned}$$

This contradicts the preconditions $k \in Id$ and $ed \in ED$. Hence, the assumption of $\mathsf{ddec}_i.(j, (k, ed)) \lhd tr$ is wrong and $\mathsf{ddec}_i.(j, (k, ed)) \not\lhd tr$ holds.

(b) The proof goes along the lines of the one for the previous part, with ED exchanged by CE and ddec exchanged by rdec.

(c) We show the claim by contradiction and assume that there exist $x \in CE \cup ED$ and $i, j \in Id$ with $j \neq i$, such that $\mathsf{fwd}_j.(i, (i, x)) \lhd tr$ holds.

$$\begin{aligned}
&\xRightarrow{\rho_j(\mathsf{SRP}_j)} \mathsf{rreq}_j.(i, x) \lhd tr\\
&\xRightarrow{\rho_j(\mathsf{COR}_j)} \mathsf{link}_{k,j}.(i, x) \lhd tr \text{ for some identifier } k \neq j\\
&\xRightarrow{\rho_k(\mathsf{COR}_k)} \mathsf{ddec}_k.(j,(i,x)) \lhd tr \vee \mathsf{rdec}_k.(j,(i,x)) \lhd tr \vee \mathsf{fwd}_k.(j,(i,x)) \lhd tr\\
&\quad \xRightarrow{(*)} \mathsf{ddec}_k.(j,(i,x)) \lhd tr \upharpoonright \alpha(\rho_k(\mathsf{SRP}_k)) \wedge tr \upharpoonright \alpha(\rho_k(\mathsf{SRP}_k)) \in traces(\rho_k(\mathsf{SRP}_k))\\
&\qquad \vee \mathsf{rdec}_k.(j,(i,x)) \lhd tr \upharpoonright \alpha(\rho_k(\mathsf{SRP}_k)) \wedge tr \upharpoonright \alpha(\rho_k(\mathsf{SRP}_k)) \in traces(\rho_k(\mathsf{SRP}_k))\\
&\qquad \vee \mathsf{fwd}_k.(j,(i,x)) \lhd tr \upharpoonright \alpha(\rho_k(\mathsf{SRP}_k)) \wedge tr \upharpoonright \alpha(\rho_k(\mathsf{SRP}_k)) \in traces(\rho_k(\mathsf{SRP}_k))\\
&\quad \Longrightarrow \text{by definition of } \mathsf{SRP}_k\\
&\qquad j = nxt(k, i)\\
&\quad \Longrightarrow \text{by definition of } nxt \text{ [1, page 12]}\\
&\qquad j = nxt(k, i) = i
\end{aligned}$$

The last equation contradicts the initial assumption of $i \neq j$. Hence, the assumption $\mathsf{fwd}_j.(i, (i, x)) \lhd tr$ cannot hold, i.e., $\mathsf{fwd}_j.(i, (i, x)) \not\lhd tr$ holds for all $j \neq i$.

(d) We show the claim by contradiction and assume there exists an identifier $i \in Id$ and a decision $ed = (ev, ev') \in CE_i^{\mathsf{Id}} \times CE_i$ such that $\mathsf{appv}_i.ed \lhd tr$ holds.

$$\begin{aligned}
&\xRightarrow{\rho_i(\mathsf{SRP}_i)} \mathsf{rreq}_i.(i, ed) \lhd tr \xRightarrow{\rho_i(\mathsf{COR}_i)} \mathsf{link}_{j,i}.(i, ed) \lhd tr \text{ for some identifier } j \neq i\\
&\xRightarrow{\rho_j(\mathsf{COR}_j)} \mathsf{ddec}_j.(i,(i,ed)) \lhd tr \vee \mathsf{fwd}_j.(i,(i,ed)) \lhd tr \vee \mathsf{rdec}_j.(i,(i,ed)) \lhd tr\\
&\quad \Longrightarrow \text{by Lemma 3 (a) and Lemma 3 (c)}\\
&\qquad \mathsf{rdec}_j.(i,(i,ed)) \lhd tr\\
&\xRightarrow{\rho_j(\mathsf{SRP}_j)} \mathsf{rtrsp}_j.(i, ed) \lhd tr \xRightarrow{\rho_j(\mathsf{DEC}_j(\emptyset))} \mathsf{rereq}_j.ev \lhd tr \xRightarrow{\rho_j(\mathsf{SRP}_j)} \mathsf{rreq}_j.(j, ev) \lhd tr\\
&\xRightarrow{\rho_j(\mathsf{COR}_j)} \mathsf{link}_{k,j}.(j, ev) \lhd tr \text{ for some identifier } k \neq j\\
&\xRightarrow{\rho_k(\mathsf{COR}_k)} \mathsf{ddec}_k.(j,(j,ev)) \lhd tr \vee \mathsf{fwd}_k.(j,(j,ev)) \lhd tr \vee \mathsf{rdec}_k.(j,(j,ev)) \lhd tr
\end{aligned}$$

$\Longrightarrow$ by Lemma 3 (b) and Lemma 3 (c)

$\mathsf{ddec}_k.(j,(j,ev)) \lhd tr$

$\xRightarrow{\rho_k(\mathsf{SRP}_k)}$ $\mathsf{rtreq}_k.(j,ev) \lhd tr$

$\xRightarrow{(*)}$ $\mathsf{rtreq}_k.(j,ev) \lhd tr \upharpoonright \alpha(\rho_k(\mathsf{DEL}_k)) \wedge tr \upharpoonright \alpha(\rho_k(\mathsf{DEL}_k)) \in traces(\rho_k(\mathsf{DEL}_k))$

$\Longrightarrow$ by definition of DEL_k, according to which $\mathsf{rtreq}.(j,ev)$ can only occur if $ev \in \{ev' \in CE_k \mid k \neq resp(ev')\}$ holds

$ev \in CE_k \wedge k \neq resp(ev)$

$\Longrightarrow$ by definition of id [1, page 11]

$id(ev) \neq resp(ev)$

$\Longrightarrow$ by definition of CE_i^{ld}

$ev \notin CE_i^{\mathrm{ld}}$

The final consequence contradicts the initial assumption of $ed = (ev, ev') \in CE_i^{\mathrm{ld}} \times CE_i$. Hence, the assumption $\mathsf{appv}_i.ed \lhd tr$ is wrong and $\mathsf{appv}_i.ed \ntriangleleft tr$ holds.

(e) Let identifier $i \in Id$ and decision $ed = (ev, ev') \in ED_i \setminus (CE_i^{\mathrm{ld}} \times CE_i)$ be arbitrary but fixed. We show the claim by contradiction and assume $\mathsf{edec}_i.ed \lhd tr$ holds.

$\xRightarrow{\rho_i(\mathsf{DEC}_i(\emptyset))}$ $\mathsf{lereq}_i.ev \lhd tr$

$\xRightarrow{(*)}$ $\mathsf{lereq}_i.ev \lhd tr \upharpoonright \alpha(\rho_i(\mathsf{DEL}_i)) \wedge tr \upharpoonright \alpha(\rho_i(\mathsf{DEL}_i)) \in traces(\rho_i(\mathsf{DEL}_i))$

$\Longrightarrow$ by definition of DEL_i, according to which $\mathsf{lereq}.ev$ can only occur if $ev \in \{ev' \in CE_i \mid i = resp(ev')\}$

$i = resp(ev)$

$\Longrightarrow$ by the precondition that $ev \in CE_i$ and by definition of CE_i^{ld}

$ev \in CE_i^{\mathrm{ld}}$

The final statement contradicts the precondition $ed = (ev, ev') \in ED_i \setminus (CE_i^{\mathrm{ld}} \times CE_i)$, from which $ev \notin CE_i^{\mathrm{ld}}$ follows immediately. □

Lemma 4. *For all events $ev \in CE$ and traces $tr \in traces(\mathsf{SI})$ with $ev \lhd tr$, it holds that there exists an event $ev' \in CE_{id(ev)}$ with $\mathsf{enf}_{id(ev)}.(ev', ev) \lhd tr$.*

Proof. Let $ev \in CE$ be an event and $tr \in traces(\mathsf{SI})$ be a trace with $ev \lhd tr$. Let $i := id(ev)$.

$\xRightarrow{(*)}$ with $ev \in CE_{id(ev)} = CE_i$ by definition of id [1, page 11], $CE_i \subseteq \alpha(\mathsf{REPLACE}_i)$ by definition of $\mathsf{REPLACE}_i = \mathsf{REPLACE}(CE_i)$ [1, page 8], and $\rho_i(ev) = ev$ according to Definition 1 (b)

$ev \lhd tr \upharpoonright \alpha(\rho_i(\mathsf{REPLACE}_i)) \wedge tr \upharpoonright \alpha(\rho_i(\mathsf{REPLACE}_i)) \in traces(\rho_i(\mathsf{REPLACE}_i))$

$\xRightarrow{(**)}$ by the precondition that CE_i, defined in [1, page 11], is disjoint from the set $\{\mathsf{sync}.\checkmark, \mathsf{enf}.(ev', ev) \mid ev, ev' \in CE_i\}$

$\mathsf{enf}_i.(ev', ev) \lhd tr \upharpoonright \alpha(\rho_i(\mathsf{REPLACE}_i))$ for some event $ev' \in CE_i$

$\Longrightarrow$ with $i = id(ev)$

$\mathsf{enf}_{id(ev)}.(ev', ev) \lhd tr$

Note that the second implication holds, because we concretized the replacement sequences EA as CE_i (i.e., sequences of length 1 of critical events) in [1, page 12]. Therefore, a simplified equivalent specification of the replacer process is

$$\mathsf{REPLACE}_i = \mathsf{enf}?(ev', ev)\colon CE_i \times CE_i \to ev \to \mathsf{sync}!\checkmark \to \mathsf{REPLACE}_i \qquad \Box$$

Lemma 5. *Let $ev \in CE \setminus DUM$ be an event and $tr \in traces(\mathsf{SI})$ with $ev \lhd tr$ be a trace. Then $\mathsf{edec}_{resp(ev)}.(ev, ev) \lhd tr$ or $\mathsf{rtrsp}_{resp(ev)}.(id(ev), (ev, ev)) \lhd tr$ holds true.*

Proof. Let $ev \in CE \setminus DUM$ be an event and $tr \in traces(\mathsf{SI})$ be a trace with $ev \lhd tr$. Let $i := id(ev)$.

$$\Longrightarrow \text{ by Lemma 4}$$

$$\mathsf{enf}_i.(ev', ev) \lhd tr \text{ for some event } ev' \in CE_i$$

$$\xRightarrow{\rho_i(\mathsf{COR}_i)} \mathsf{edec}_i.(ev', ev) \lhd tr \lor \mathsf{appv}_i.(ev', ev) \lhd tr$$

$$\Longrightarrow \text{ by Lemma 3 (d) and Lemma 3 (e)}$$

$$(\mathsf{edec}_i.(ev', ev) \lhd tr \land id(ev') = resp(ev')) \lor (\mathsf{appv}_i.(ev', ev) \lhd tr \land id(ev') \neq resp(ev'))$$

In the following, we show the claim for both disjuncts separately.

(a)

$$\mathsf{edec}_i.(ev', ev) \lhd tr \land id(ev') = resp(ev')$$

$$\xRightarrow{(*)} \mathsf{edec}_i.(ev', ev) \lhd tr \upharpoonright \alpha(\rho_i(\mathsf{DEC}_i(\emptyset))) \land tr \upharpoonright \alpha(\rho_i(\mathsf{DEC}_i(\emptyset))) \in traces(\rho_i(\mathsf{DEC}_i(\emptyset)))$$

$$\Longrightarrow \text{ by definition of } \mathsf{DEC}_i(\emptyset)$$

$$\mathsf{edec}_i.(ev', ev) \lhd tr \land (ev' = ev \lor ev = dummy_{ev'})$$

$$\Longrightarrow \text{ by the preconditions } ev \in CE \setminus DUM \text{ and } dummy_{ev'} \in DUM$$

$$\mathsf{edec}_i.(ev', ev) \lhd tr \land ev = ev'$$

$$\Longrightarrow \text{ by the preconditions } i = id(ev) \text{ and } id(ev') = resp(ev')$$

$$\mathsf{edec}_{resp(ev)}.(ev, ev) \lhd tr$$

(b)

$$\mathsf{appv}_i.(ev', ev) \lhd tr \land id(ev') \neq resp(ev')$$

$$\xRightarrow{\rho_i(\mathsf{SRP}_i)} \mathsf{rreq}_i.(i, (ev', ev)) \lhd tr$$

$$\xRightarrow{\rho_i(\mathsf{COR}_i)} \mathsf{link}_{j,i}.(i, (ev', ev)) \lhd tr \text{ for some identifier } j \neq i$$

$$\xRightarrow{\rho_j(\mathsf{COR}_j)} \mathsf{ddec}_j.(i, (i, (ev', ev))) \lhd tr \lor \mathsf{fwd}_j.(i, (i, (ev', ev))) \lhd tr \lor \mathsf{rdec}_j.(i, (i, (ev', ev))) \lhd tr$$

$$\Longrightarrow \text{ by Lemma 3 (a) and Lemma 3 (c)}$$

$$\mathsf{rdec}_j.(i, (i, (ev', ev))) \lhd tr$$

$$\xRightarrow{\rho_j(\mathsf{SRP}_j)} \mathsf{rtrsp}_j.(i, (ev', ev)) \lhd tr \qquad (\dagger)$$

$$\xRightarrow{(*)} \mathsf{rtrsp}_j.(i, (ev', ev)) \lhd tr \upharpoonright \alpha(\rho_j(\mathsf{DEC}_j(\emptyset))) \land tr \upharpoonright \alpha(\rho_j(\mathsf{DEC}_j(\emptyset))) \in traces(\rho_j(\mathsf{DEC}_j(\emptyset)))$$

$$\xRightarrow{(**)} \text{ by definition of } \mathsf{DEC}_j(\emptyset)$$

$$\mathsf{rereq}_j.ev' \lhd tr \upharpoonright \alpha(\rho_j(\mathsf{DEC}_j(\emptyset))) \land (ev = ev' \lor ev = dummy_{ev'})$$

$$\Longrightarrow \text{ by the preconditions } ev \in CE \setminus DUM \text{ and } dummy_{ev'} \in DUM$$

$$\mathsf{rereq}_j.ev' \lhd tr \restriction \alpha(\rho_j(\mathsf{DEC}_j(\emptyset))) \land ev = ev' \qquad (\ddagger)$$
$$\Longrightarrow \mathsf{rereq}_j.ev \lhd tr$$
$$\xRightarrow{\rho_j(\mathsf{SRP}_j)} \mathsf{rreq}_j.(j, ev) \lhd tr$$
$$\xRightarrow{\rho_j(\mathsf{COR}_j)} \mathsf{link}_{k,j}.(j, ev) \lhd tr \text{ for some identifier } k \neq j$$
$$\xRightarrow{\rho_k(\mathsf{COR}_k)} \mathsf{ddec}_k.(j,(j,ev)) \lhd tr \lor \mathsf{fwd}_k.(j,(j,ev)) \lhd tr \lor \mathsf{rdec}_k.(j,(j,ev)) \lhd tr$$
$$\Longrightarrow \boxed{\text{by Lemma 3 (b) and Lemma 3 (c)}}$$
$$\mathsf{ddec}_k.(j,(j,ev)) \lhd tr$$
$$\xRightarrow{\rho_k(\mathsf{SRP}_k)} \mathsf{rtreq}_k.(j, ev) \lhd tr$$
$$\xRightarrow{(*)} \mathsf{rtreq}_k.(j, ev) \lhd tr \restriction \alpha(\rho_k(\mathsf{DEL}_k)) \land tr \restriction \alpha(\rho_k(\mathsf{DEL}_k)) \in \mathit{traces}(\rho_k(\mathsf{DEL}_k))$$
$$\Longrightarrow \boxed{\text{by definition of } \mathsf{DEL}_k}$$
$$j = \mathit{resp}(ev)$$
$$\Longrightarrow \boxed{\text{with } \mathsf{rtrsp}_j.(i,(ev',ev)) \lhd tr \text{ from } (\dagger) \text{ above, } ev = ev' \text{ from } (\ddagger) \text{ above, and precondition } i = \mathit{id}(ev)}$$
$$\mathsf{rtrsp}_{\mathit{resp}(ev)}.(\mathit{id}(ev),(ev,ev)) \lhd tr \qquad \square$$

Lemma 6. *Let trace $tr \in \mathit{traces}(\mathsf{SI})$, identifier $i \in \mathit{Id}$, and events $ev, ev' \in \mathit{CE} \setminus \mathit{DUM}$ with $ev \otimes ev'$ be given. Then at least one of the following holds:*

- *$\mathsf{edec}_i.(ev, ev) \not\lhd tr$ and for all identifiers $j \in \mathit{Id}$, $\mathsf{rtrsp}_i.(j,(ev,ev)) \not\lhd tr$, or*
- *$\mathsf{edec}_i.(ev', ev') \not\lhd tr$ and for all identifiers $j \in \mathit{Id}$, $\mathsf{rtrsp}_i.(j,(ev',ev')) \not\lhd tr$.*

Proof. Let trace $tr \in \mathit{traces}(\mathsf{SI})$, identifier $i \in \mathit{Id}$, and events $ev, ev' \in \mathit{CE} \setminus \mathit{DUM}$ with $ev \otimes ev'$ be arbitrary but fixed. Let $D_x := \{\mathsf{edec}_i.(x,x), \mathsf{rtrsp}_i.(j,x,x) \mid j \in \mathit{Id}\}$ for all $x \in \{ev, ev'\}$. We show that $tr \restriction D_{ev} = \langle\rangle$ or $tr \restriction D_{ev'} = \langle\rangle$ holds, which is equivalent to $(tr \restriction D_{ev} \neq \langle\rangle) \implies (tr \restriction D_{ev'} = \langle\rangle)$. Thus, we assume $tr \restriction D_{ev} \neq \langle\rangle$ and show $tr \restriction D_{ev'} = \langle\rangle$.

$$tr \restriction D_{ev} \neq \langle\rangle$$
$$\iff \exists e \in D_{ev}.(e \lhd tr)$$
$$\xRightarrow{(*)} e \lhd tr \restriction \alpha(\rho_i(\mathsf{DEC}_i(\emptyset))) \land tr \restriction \alpha(\rho_i(\mathsf{DEC}_i(\emptyset))) \in \mathit{traces}(\rho_i(\mathsf{DEC}_i(\emptyset)))$$
$$\iff \boxed{\text{with } tr_{\text{dec}} := tr \restriction \alpha(\rho_i(\mathsf{DEC}_i(\emptyset)))}$$
$$e \lhd tr_{\text{dec}} \land tr_{\text{dec}} \in \mathit{traces}(\rho_i(\mathsf{DEC}_i(\emptyset)))$$
$$\xRightarrow{(a)} e \lhd tr_{\text{dec}} \land tr_{\text{dec}} \in \mathit{traces}(\rho_i(\mathsf{DEC}_i(\emptyset))) \land e \notin D_{ev'}$$
$$\Longrightarrow \boxed{\text{without loss of generality}}$$
$$tr \restriction \alpha(\rho_i(\mathsf{DEC}_i(\emptyset))) = s_1.\langle e\rangle.s_2 \land s_1 \restriction D_{ev'} = \langle\rangle$$
$$\xRightarrow{(b)} \exists q \in 2^{\mathit{CE}}.(ev \in q \land \rho_i(\mathsf{DEC}_i(\emptyset)) / (s_1.\langle e\rangle) = \rho_i(\mathsf{DEC}_i(q)))$$
$$\Longrightarrow \boxed{\text{by Lemma 1 (e)}}$$
$$s_2 \in \mathit{traces}(\rho_i(\mathsf{DEC}_i(q))) \land ev \in q$$
$$\xRightarrow{(c)} s_2 \restriction D_{ev'} = \langle\rangle$$
$$\Longrightarrow \boxed{\text{with } s_1 \restriction D_{ev'} = \langle\rangle \text{ and } e \notin D_{ev'} \text{ from above}}$$
$$tr_{\text{dec}} \restriction D_{ev'} = (s_1.\langle e\rangle.s_2) \restriction D_{ev'} = \langle\rangle$$

$$\Longrightarrow \boxed{\text{by preconditions } D_{ev'} \subseteq \alpha(\rho_i(\mathsf{DEC}_i(\emptyset))) \text{ and } tr_{\text{dec}} = tr \restriction \alpha(\rho_i(\mathsf{DEC}_i(\emptyset)))}$$
$$tr \restriction D_{ev'} = \langle\,\rangle$$

Below, we prove the correctness of the implications labeled (a), (b), and (c) above.

(a) We show that $e \notin D_{ev'}$ follows from $e \lhd tr_{\text{dec}} \wedge tr_{\text{dec}} \in \mathit{traces}(\rho_i(\mathsf{DEC}_i(\emptyset)))$.

$$e \lhd tr_{\text{dec}} \wedge tr_{\text{dec}} \in \mathit{traces}(\rho_i(\mathsf{DEC}_i(\emptyset)))$$
$$\Longrightarrow \boxed{\text{by definition of } \mathsf{DEC}_i(\emptyset)) \text{ and the preconditions } e \in D_{ev} \text{ and } ev \notin \mathit{DUM}}$$
$$\exists q \in 2^{CE}.(ev \notin \mathit{conf}(q))$$
$$\Longrightarrow \boxed{\text{by definition of } \mathit{conf} \text{ [1, page 12]}}$$
$$\neg(ev \otimes ev)$$
$$\Longrightarrow \boxed{\text{by precondition } ev \otimes ev'}$$
$$ev \neq ev'$$
$$\Longrightarrow \boxed{\text{by definition of } D_{ev'}}$$
$$e \notin D_{ev'}$$

(b) We show that there exists a state $q \in 2^{CE}$ with $ev \in q$ such that $\rho_i(\mathsf{DEC}_i(\emptyset)) \,/\, (s_1.\langle e\rangle) = \rho_i(\mathsf{DEC}_i(q))$ holds. By definition of $\rho_i(\mathsf{DEC}_i(\emptyset))$, this process only engages in e (recall firstly that e must be one of $\mathsf{edec}_i.(ev, ev)$ and $\mathsf{rtrsp}_i.(j, ev, ev)$ for some $j \in \mathit{Id}$, and secondly that $ev \notin \mathit{DUM}$) if it afterwards immediately adds ev to its previously active state, say $q' \in 2^{CE}$, and then behaves as $\rho_i(\mathsf{DEC}_i(q))$ for $q = q' \cup \{ev\}$.

(c) We show by induction over the length of traces s that for all states $q' \in 2^{CE}$ with $ev \in q'$, it holds that $s \in \mathit{traces}(\rho_i(\mathsf{DEC}_i(q')))$ implies $s \restriction D_{ev'} = \langle\,\rangle$.

base case ($|s| \leq 1$): By definition of $\mathsf{DEC}_i(q')$, events from $D_{ev'}$ cannot be contained in such a short trace s.

step case ($s = s_1.s_2$ for $|s_1| = 2$): For all $q' \in 2^{CE}$, the definition of $\mathsf{DEC}_i(q')$ gives:

- An event $e' \in D_{ev'}$ cannot be contained in s_1 because of the definition of *conf* [1, page 12] and the precondition $ev' \otimes ev$; hence, we have $s_1 \restriction D_{ev'} = \langle\,\rangle$.
- $\mathsf{DEC}_i(q') \,/\, s_1 = \mathsf{DEC}_i(q'')$ for $q'' \supseteq q'$; hence, the induction hypothesis can be applied on q'' and s_2 to obtain $s_2 \restriction D_{ev'} = \langle\,\rangle$.

It follows that $s \restriction D_{ev'} = \langle\,\rangle$. □

Based on the preceding lemmas and the auxiliary definition, we can now conduct the proof that the controlled system defined in [1, Section 5] enforces the Chinese Wall policy, which it is instantiated to enforce.

Proof of Theorem 1. We show that $\mathsf{SYSTEM}\,\mathsf{sat}\,\mathit{ChW}$ holds. Substituting the definitions of sat and *ChW*, this is equivalent to proving that for all traces $tr \in \mathit{traces}(\mathsf{SYSTEM})$ there do not exist events $ev_1, ev_2 \in CE$ such that $ev_1, ev_2 \lhd tr$ and $ev_1 \otimes ev_2$ hold.

We conduct the proof by contradiction and assume that $\mathsf{SYSTEM}\,\mathsf{sat}\,\mathit{ChW}$ does not hold. It follows that there exist a trace $tr \in \mathit{traces}(\mathsf{SYSTEM})$ and events $ev_1, ev_2 \in CE$ such that $ev_1, ev_2 \lhd tr$, and $ev_1 \otimes ev_2$ hold true. Since events ev_1 and ev_2 are in conflict, they cannot be dummy events (by definition of dummy events [1, page 11]). Secondly, the responsible node for ev_1 and ev_2 must be the same, i.e., $\mathit{resp}(ev_1) = \mathit{resp}(ev_2)$ (by the definition of *resp* [1, page 12]). Let $k := \mathit{resp}(ev_1)$ be this responsible node.

In the following, the process SI (see Definition 1 (d)) denotes the instantiated service automata framework with all CSP hiding operations removed and the internal channels renamed to not introduce additional synchronization with the removal of hiding. We assume, without loss of generality, that the service and data providers PROV_i do not make use of local policy-internal channels and the renamed channels either (if they would, we could choose a different renaming). Then by Lemma 2, we obtain $\mathsf{SI} \setminus HS = \mathsf{SYSTEM}$, where the hiding set HS is defined in Definition 1 (c). It follows that there must be a trace $tr' \in traces(\mathsf{SI})$ such that $tr' \upharpoonright \alpha(\mathsf{SYSTEM}) = tr$. Particularly, we therefore have $ev_1, ev_2 \lhd tr'$.

Applying Lemma 5 for each event $ev \in \{ev_1, ev_2\}$, we get that in both cases either event $\mathsf{edec}_k.(ev, ev)$ or event $\mathsf{rtrsp}_k.(id(ev), ev, ev)$ is contained in tr'. This contradicts Lemma 6, which states that this can only hold for at most one of ev_1 and ev_2. Consequently, the assumption that $\mathsf{SYSTEM}\,\mathsf{sat}\,\mathit{ChW}$ does not hold leads to a contradiction and, thus, cannot be true. Therefore, $\mathsf{SYSTEM}\,\mathsf{sat}\,\mathit{ChW}$ holds. □

References

[1] Richard Gay, Heiko Mantel, and Barbara Sprick. Service Automata. In Gilles Barthe, Anupam Datta, and Sandro Etalle, editors, *Proceedings of the 8th International Workshop on Formal Aspects of Security and Trust*, 2011. In press.

[2] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, Inc., 1985.