Information Flow Security for Service Compositions*

Sabina Rossi and Damiano Macedonio
Università Ca' Foscari, Dipartimento di Informatica, via Torino 155, 30172 Venice, Italy
e-mail:{srossi, mace}@dsi.unive.it

Abstract

Web services are ubiquitous technologies which are used for integrating business processes and services. As is the case in many other applications, the information processed in web services might be commercially sensitive and it is important to protect this information against security threats such as disclosure to unauthorized parties.

In this paper we propose a notion of non-interference *for service compositions expressed as terms of a typed CCS-like process algebra. Security policies are used to specify the security requirements of service components and may be dynamically provided by the service participants.*

We study the conditions under which service components may be replaced in a service composition while preserving both non-interference and compliance, that is a basic property ensuring the absence of livelocks and deadlocks during execution.

Keywords: Process algebra, behavioural equivalences, non-interference, web services

1 Introduction

Service Oriented Architectures (SOA) provide a software architectural style to connect loosely specied and coupled services that communicate with each other. Web services are the most prominent software systems used to realize SOA. They are designed to support interoperable service-to-service interactions over a network. This interoperability is gained through a set of XML-based open standards, such as WSDL, SOAP, and UDDI. These standards provide a common approach for defining, publishing, and using web services.

Web services are mostly used for integrating business processes and services. As is the case in many other applications, the information processed in web services might be commercially sensitive and it is important to protect this information against security threats such as disclosure to unauthorized parties. Indeed, many of the features that make web services attractive, including greater accessibility of data, dynamic connections, and relative autonomy (lack of human intervention) are at odds with traditional security models and controls. Difficult issues and unsolved problems exist, such as protecting confidentiality and integrity of data that is transmitted through web services protocols in service-to-service transactions.

Traditional network security technologies (e.g., firewalls) are inadequate to protect SOAs. Indeed SOAP messages are transmitted over HTTP, which is allowed to flow without restriction through proxies and firewalls. Moreover, the protocol SSL/TLS, which is used to authenticate and encrypt web-based messages, is inadequate for protecting SOAP messages because it is designed to operate between two endpoints. SSL/TLS cannot accommodate web services' inherent ability to forward messages to multiple other web services. The web service processing model requires the ability to secure SOAP messages and XML documents as they are forwarded along potentially long chains of consumers, providers, and other intermediary services.

In this paper we propose a notion of *non-interference* [13] for multilevel service compositions. This is motivated by the fact that SOAs are increasingly relying on complex distributed systems that share information with multiple levels of security. In these systems information with mixed security levels is processed and targeted to particular clients. For example, in a e-business system, some data will be privileged (e.g., credit card numbers and medical records) and some data will be public (e.g., stock market quotes). Such systems need to be equipped with appropriate security facilities to guarantee the security requirements (e.g., confidentiality or integrity) of the participants.

Traditionally, security policies are used to specify the security requirements of a system and to control the access to confidential data and resources (see, e.g., [1, 3]). However, if this approach can be successfully applied to enforce the security requirements of a centralized system, it is less suited to formulate the security needs of a service composition. Indeed, there is no central authority in the web that

*Corresponding author: Sabina Rossi.

is able to fix the security labels of all services and data. Rather than manipulating stored data, web services computes requested information from dynamic data available on the net that need to be dynamically classified according to their stored information.

We present an information flow security model [20, 15, 12, 19] for service compositions to control the flow of confidential data in web services. Our framework is based on the use of security policies which may be dynamically provided by the service components.

We specify service compositions in terms of behavioral contracts which provide abstract descriptions of system behaviors by means of terms of some process algebra. Formal theories of contracts have first been introduced in [8], and then further developed along independent lines of research in [9, 10, 14], and independently in [4, 5].

Our model is based on the notion of *non-interference* [13, 11, 18] and is inspired by our previous work [12] for CCS processes. The definition of non-interference for service compositions we present here demands that public synchronizations are unchanged as confidential communications are varied or, more generally, that the low level behaviour of the composition is independent from the behaviour of its high components. Security policies are used to specify the security requirements of service components. In order to capture the dynamic nature, heterogeneity and lack of knowledge which are intrinsic features of modern web services, we allow policies to be dynamically specified by the service participants. In our model, for example, customers may formulate their security requirements by dynamically assign types (that are security annotations) to individual service components.

Moreover, we study the conditions under which service components may be safely replaced in a service composition while preserving both non-interference and compliance [2, 7], that is a property ensuring the absence of livelocks and deadlocks during execution. We also provide a coinductive proof technique for the safe replacement of single components in multy-party service compositions.

The paper is organized as follows: Section 2 introduces the contract-based language we use to specify multilevel service compositions. Section 3 formalizes the notion of non-interference. Section 4 studies the conditions under which a single contract can be safely replaced in a service composition while preserving both compliance and non-interference. Finally, Section 5 concludes the presentation.

2 A Process Algebra for Multilevel Service Compositions

We represent service contracts as terms of a value-passing CCS-like [16] process calculus that includes recursion and operators for external and internal choice. Parallel composition arises in *contract compositions* that we define as the parallel (and concurrent) composition of a set of principals executing contracts. We presuppose a denumerable set of action names $\mathcal{A}$, ranged over by a, b, c, a denumerable set of principal identities $\mathcal{P}$, ranged over by p, q, r, and a denumerable set of variables $\mathcal{V}$, ranged over by x, y. The actions represent the basic units of observable behavior of the underlying services, while the principal names specify the peers providing the services.

In order to specify multilevel service compositions, we assign security levels to principal identities and express contracts and compositions as typed terms of our process calculus. Formally, we assume a complete lattice $\langle \Sigma, \preceq \rangle$ of security annotations, ranged over by ς, ϱ, where $\top$ and $\bot$ represent the top and the bottom elements of the lattice. We denote by $\sqcup$ and $\sqcap$, respectively, the join and meet operators over Σ. Type environments are used to assign security levels to principals. A type environments Γ is a finite mapping from principals and variables to security annotations. We define $\Gamma_1 \sqcup \Gamma_2$ (resp. $\Gamma_1 \sqcap \Gamma_2$) the type environment Γ such that $\Gamma(p) = \Gamma_i(p)$ if $p \notin dom(\Gamma_1) \cap dom(\Gamma_2)$ and $p \in dom(\Gamma_i)$, otherwise $\Gamma(p) = \Gamma_1(p) \sqcup \Gamma_2(p)$ (resp. $\Gamma(p) = \Gamma_1(p) \sqcap \Gamma_2(p)$).

The syntax of our calculus is presented in Table 1.

The term $\mathbf{1}$ indicates a contract that has reached a successful state. The contract $\bar{a}@p.\sigma$ describes a service that sends a message on a to principal p and then behaves as σ; syntactically, the principal identity p may be a variable, but it must be a name when the prefix is ready to fire. Dually, the input prefix $a@u.\sigma$ waits for an input on a from a particular/any principal and then continues as σ. If u is a variable x, then the input form is a binder for x with scope σ: upon synchronization with a principal p, x gets uniformly substituted by p in σ. The contract $\sigma + \sigma'$ denotes an external choice, guided by the environment. The contract $\sigma \lfloor_{\Gamma} \oplus {}_{\Gamma'} \rfloor \sigma'$ represents the internal choice between σ in the type environment Γ and σ' in the type environment Γ' made irrespective of the structure of the interacting components; the internal choice operator we adopt in this paper allows us to model the fact that a principal may dynamically change (upgrade) the security level of his interactions with other service components through an internal choice. Finally, $\texttt{rec}(\mathbf{x})\,\sigma$ makes it possible to express iteration in the contract language. As usual, we assume a standard contractivity condition for the recursion operator, requiring that recursion variables be guarded by a prefix.

Given a principal $p \in \mathcal{P}$, we say that a contract σ is *p-compatible* if for all $\bar{a}@q$ and $a@q$ occurring in σ, q is different from p.

A composition $p_1[\sigma_1] \;\parallel\; \cdots \;\parallel\; p_n[\sigma_n]$ of principals must be *well-formed* [5, 6] to constitute a legal composition, namely: (i) the principal identities p_i's must all be pairwise

Table 1 Syntax

Type environments	Γ	$::=$	$\emptyset \mid \Gamma, p:\varsigma$	$p \in \mathcal{P},\ \varsigma \in \Sigma$
Actions	φ	$::=$	$\bar{a}@u \mid a@u$	$a \in \mathcal{A},\ u \in \mathcal{P} \cup \mathcal{V}$
Contracts	σ	$::=$	$\mathbf{1} \mid \mathbf{x} \mid \varphi.\sigma \mid \sigma + \sigma \mid \sigma \lfloor_{\Gamma}\oplus\ _{\Gamma}\rfloor \sigma \mid \mathtt{rec}(\mathbf{x})\,\sigma$	
Compositions	C	$::=$	$p[\sigma] \mid C \parallel C$	

Table 2 Example of a travel agency

$$S = C[\sigma_C] \parallel T[\sigma_T] \parallel A_1[\sigma_A] \parallel A_2[\sigma_A]$$

$$\sigma_C = \overline{\mathtt{Req}}@T.\mathtt{Lst}@T.(\,\overline{\mathtt{Close}}@T.\mathbf{1}\lfloor_{\emptyset}\oplus\ _{T:\mathtt{H}}\rfloor(\,\overline{\mathtt{Buy1}}@T.\overline{\mathtt{Pay}}@T.\mathtt{Get}@A_1.\mathbf{1}\lfloor_{A_1:\mathtt{H}}\oplus\ _{A_2:\mathtt{H}}\rfloor\overline{\mathtt{Buy2}}@T.\overline{\mathtt{Pay}}@T.\mathtt{Get}@A_2.\mathbf{1}\,)\,)$$

$$\sigma_T = \mathtt{Req}@x.\overline{\mathtt{Inq}}@A_1.\overline{\mathtt{Inq}}@A_2.\mathtt{Price}@A_1.\mathtt{Price}@A_2.\overline{\mathtt{Lst}}@x.(\,\mathtt{Close}@x.\mathbf{1} + \mathtt{Buy1}@x.\overline{\mathtt{Ord}}@A_1.\mathtt{Pay}@x.\overline{\mathtt{Conf}}@A_1.\mathbf{1} + \mathtt{Buy2}@x.\overline{\mathtt{Ord}}@A_2.\mathtt{Pay}@x.\overline{\mathtt{Conf}}@A_2.\mathbf{1}\,)$$

$$\sigma_A = \mathtt{Inq}@x.\overline{\mathtt{Price}}@x.(\,\mathtt{Ord}@x.\mathtt{Conf}@x.\overline{\mathtt{Get}}@y.\mathbf{1} + \mathbf{1}\,)$$

different, and (ii) each contract σ_i, executed by principal p_i, is p_i-compatible. If $C = p_1[\sigma_1] \parallel \cdots \parallel p_n[\sigma_n]$ is a legal composition, we say that C is a $\{p_1, \ldots, p_n\}$-composition (dually, that $\{p_1, \ldots, p_n\}$ are the underlying principals for C).

Throughout, we assume that contracts are closed (they have no free variables) and that compositions are well formed. Also, we often omit trailing $\mathbf{1}$'s.

A service component may modify the security level of its interactions with other components by assigning different security levels to the principals with which it is going to interact. However, it is reasonable to assume that a service component cannot downgrade the security level of other principals; moreover it cannot upgrade the level of its interactions with other components above its own level. These are the only typing constraints we assume. Such a typing discipline ensures that information does not explicitly flow from high to low, but it does not deal with implicit flows. Instead, we characterize non-interference in terms of the actions that typed service compositions may perform.

Example 2.1 Table 2 shows an example of a service contract composition. Four services are involved in this composition: $C[\sigma_C]$, $T[\sigma_T]$ and $A_i[\sigma_A]$ representing a *customer*, a *travel agency*, and two *airline companies*, respectively. The elementary actions represent business activities that result in messages being sent or received. For example, the action $\overline{\mathtt{Req}}@T$ undertaken by the customer results in a message being sent to the travel agency. In the example, the customer sends a request to the travel agency which then inquires the airlines to get the prices for the selected route. Each airline responds and the travel agency sends to the customer the list of the best prices. The customer decides whether to close the interaction with the travel agency or to buy from one of the airlines. In the latter case the customer decides to assign a high security level (H) to both the travel agency and the chosen airline company in order to safeguard the confidentiality of the purchasing data. The travel agency orders the ticket from the selected airline and takes a deposit (or a full payment) from the customer. As soon as the airline receives the confirmation of the payment, the ticket is issued to the customer. □

Type System The typing rules reported in Table 3 ensure that, given a service composition with underling set of principals π, every $p \in \pi$ cannot upgrade the security level of the other principals in π (including p) above p's level.

The judgments take the form $\Gamma \vdash C$, where Γ is a type environment and C is a service composition. We denote by $\coprod$ the function that associates to each contract σ the join of all the Γ_i occurring as a parameter of an internal choice in σ. We say that a service component $p[\sigma]$ is well-typed in Γ if p will never upgrade the security level of other principals over his own level; this is obtained by requiring that for all $q \in \mathit{dom}(\coprod(\sigma))$, it holds $\coprod(\sigma)(q) \preceq \varsigma$.

Semantics of typed service compositions We define the dynamics of typed service compositions in terms of labelled transition systems (and a success predicate), with rules reported in Table 4. In the table, and in the whole paper, λ ranges over visible contract typed actions $\bar{a}@p$, $a@p$ and silent actions τ; δ ranges over service composition actions $a_{p\to q}$, $\bar{a}_{p\to q}$ and τ.

Table 3 Type System

$$\coprod(\mathbf{1}) = \emptyset \qquad \coprod(\mathbf{x}) = \emptyset \qquad \coprod(\varphi.\sigma) = \coprod(\sigma) \qquad \coprod(\sigma_1 + \sigma_2) = \coprod(\sigma_1) \sqcup \coprod(\sigma_2) \qquad \coprod(\mathtt{rec}(\mathbf{x})\,\sigma) = \coprod(\sigma)$$

$$\coprod(\sigma_1 \lfloor_{\Gamma_1} \oplus {}_{\Gamma_2} \rfloor \sigma_2) = \Gamma_1 \sqcup \Gamma_2 \sqcup \coprod(\sigma_1) \sqcup \coprod(\sigma_2)$$

$$\frac{}{\Gamma, p : \varsigma \vdash p[\sigma]}\ \varsigma \in \Sigma,\ \bigsqcup_{q \in dom(\coprod(\sigma))} \coprod(\sigma)(q) \preceq \varsigma \qquad \frac{\Gamma \vdash C_1 \quad \Gamma \vdash C_2}{\Gamma \vdash C_1 \parallel C_2}$$

Table 4 Typed contract and composition transitions

Contract and composition satisfaction: $\sigma\,\checkmark$

$$\mathbf{1}\,\checkmark \qquad \frac{\sigma_i\,\checkmark}{\sigma_1 + \sigma_2\,\checkmark} \qquad \frac{\sigma\{\mathbf{x} := \mathtt{rec}(\mathbf{x})\,\sigma\}\,\checkmark}{\mathtt{rec}(\mathbf{x})\,\sigma\,\checkmark} \qquad \frac{\sigma\,\checkmark}{p[\sigma]\,\checkmark} \qquad \frac{C_1\,\checkmark \quad C_2\,\checkmark}{C_1 \parallel C_2\,\checkmark}$$

Typed contract transitions: $\Gamma \rhd \sigma \xrightarrow{\lambda} \Gamma' \rhd \sigma'$

$$\Gamma \rhd \bar{a}@p.\sigma \xrightarrow{\bar{a}@p} \Gamma \rhd \sigma \quad \Gamma \rhd \bar{a}@x.\sigma \xrightarrow{\bar{a}@p} \Gamma \rhd \sigma\{x := p\} \quad \Gamma \rhd a@p.\sigma \xrightarrow{a@p} \Gamma \rhd \sigma \quad \Gamma \rhd a@x.\sigma \xrightarrow{a@p} \Gamma \rhd \sigma\{x := p\}$$

$$\Gamma \rhd \sigma_1 \lfloor_{\Gamma_1} \oplus {}_{\Gamma_2} \rfloor \sigma_2 \xrightarrow{\tau} \Gamma \sqcup \Gamma_i \rhd \sigma_i \quad (i = 1, 2)$$

$$\frac{\Gamma \rhd \sigma_i \xrightarrow{\lambda} \Gamma' \rhd \sigma}{\Gamma \rhd \sigma_1 + \sigma_2 \xrightarrow{\lambda} \Gamma' \rhd \sigma}\ (i = 1, 2) \qquad \frac{\Gamma \rhd \sigma\{\mathbf{x} := \mathtt{rec}(\mathbf{x})\,\sigma\} \xrightarrow{\lambda} \Gamma' \rhd \sigma'}{\Gamma \rhd \mathtt{rec}(\mathbf{x})\,\sigma \xrightarrow{\lambda} \Gamma' \rhd \sigma'}$$

Typed composition transitions: $\Gamma \rhd C \xrightarrow{\delta} \Gamma' \rhd C'$

$$\frac{\Gamma \rhd \sigma \xrightarrow{\tau} \Gamma' \rhd \sigma'}{\Gamma \rhd p[\sigma] \xrightarrow{\tau} \Gamma' \rhd p[\sigma']} \qquad \frac{\Gamma \rhd \sigma \xrightarrow{a@p} \Gamma \rhd \sigma'}{\Gamma \rhd q[\sigma] \xrightarrow{a_{p \to q}} \Gamma \rhd q[\sigma']}\ p \in dom(\Gamma),\ p \neq q \qquad \frac{\Gamma \rhd \sigma \xrightarrow{\bar{a}@p} \Gamma \rhd \sigma'}{\Gamma \rhd q[\sigma] \xrightarrow{\bar{a}_{q \to p}} \Gamma \rhd q[\sigma']}\ p \neq q$$

$$\frac{\Gamma \rhd C_1 \xrightarrow{a_{p \to q}} \Gamma \rhd C_1' \quad \Gamma \rhd C_2 \xrightarrow{\bar{a}_{p \to q}} \Gamma \rhd C_2'}{\Gamma \rhd C_1 \parallel C_2 \xrightarrow{\tau} \Gamma \rhd C_1' \parallel C_2'} \qquad \frac{\Gamma \rhd C_1 \xrightarrow{\delta} \Gamma' \rhd C_1'}{\Gamma \rhd C_1 \parallel C_2 \xrightarrow{\delta} \Gamma' \rhd C_1' \parallel C_2}$$

We say that $\Gamma \rhd C$ is a *configuration* if Γ is a type environment and C is a $\{p_1, \ldots, p_n\}$-service composition such that $\{p_1, \ldots, p_n\} \subseteq dom(\Gamma)$.

The first block of rules defines the successful states of a contract and a composition, which are those that expose the successful term $\mathbf{1}$ at top level, or immediately under an external choice (up-to recursive unfoldings). We note that a composition is a successful state only when all the components are successful.

The second block of rules defines the typed transitions for contracts, and are mostly self-explanatory. Notice that the rule for the internal choice ensures that a service component cannot downgrade the security level of other principals. Each typed contract transition yields a corresponding transition for the principal hosting the contract. The typed transitions for the configurations are relative to the underlying set $dom(\Gamma)$ of principals and are entirely standard.

We will use the following shorthands. We write $\Longrightarrow$ to note the reflexive and transitive closure of $\xrightarrow{\tau}$, and $\overset{\delta}{\Longrightarrow}$ for $\Longrightarrow\xrightarrow{\delta}\Longrightarrow$. We extend the definition to sequences of actions $w = \delta_1 \ldots \delta_n$ and we write $\overset{w}{\Longrightarrow}$ to note $\overset{\delta_1}{\Longrightarrow} \cdots \overset{\delta_n}{\Longrightarrow}$.

A computation for a configuration $\Gamma \rhd C$, is a sequence $\Gamma \rhd C = \Gamma_0 \rhd C_0 \xrightarrow{\tau} \Gamma_1 \rhd C_1 \xrightarrow{\tau} \ldots$ of internal actions.

Lemma 2.2 [Subject reduction] Let Γ be a type environment and C be a service composition such that $\Gamma \rhd C$ is a configuration. If $\Gamma \rhd C$ is well-formed and $\Gamma \rhd C \xrightarrow{\tau} \Gamma' \rhd C'$, then $\Gamma' \rhd C'$ is well formed.

Example 2.3 Consider again the service composition S of Example 2.1. Let Σ contain two security annotations, $\mathtt{L}$ (*public*) and H (*confidential*), with $\mathtt{L} \preceq \mathtt{H}$. Let Γ be the type environment $C : \mathtt{H}, T : \mathtt{L}, A_1 : \mathtt{L}, A_2 : \mathtt{L}$. The service composition S is well-typed in Γ, i.e., $\Gamma \vdash S$. $\square$

3 Non-Interference for Service Compositions

The concept of *noninterference* [13] has been introduced to formalize the absence of information flow in multilevel systems. In the context of service compositions it demands that public interactions between service components are unchanged as secret communications are varied or, more generally, that the low level behaviour of the service composition is independent from the behaviour of its high components. In this way clients are assured that the data transmitted over the air to a web server remains confidential; in other words, it cannot be intercepted and understood by eavesdroppers.

The notion of non-interference we are going to introduce is relative to the internal behaviour of service compositions, i.e., we are interested in observing the internal synchronizations between service components. We thus refine the semantics of compositions in order to help (i) to distinguish a local contract move from a synchronization, and (ii) to identify the principals involved in every synchronization. This is captured by the rules collected in Table 5, where we use the relation $\hookrightarrow$ to represent the internal typed transition of compositions. The τ action now indicates an internal action to a service component, while synchronizations between different peers in a composition are now represented through a label of the form $\{a\}_{p\to q}$ where a represents a synchronization on a and p and q are the principals executing it. We let α range over the labels $\{a\}_{p\to q}$ and τ. We define $\overset{\tau}{\hookrightarrow\!\!\!\rightarrow} \overset{def}{=} \overset{\tau}{\hookrightarrow} \cdots \overset{\tau}{\hookrightarrow}$ and $\overset{\{a\}_{p\to q}}{\hookrightarrow\!\!\!\rightarrow} \overset{def}{=} \overset{\tau}{\hookrightarrow\!\!\!\rightarrow} \cdot \overset{\{a\}_{p\to q}}{\hookrightarrow} \cdot \overset{\tau}{\hookrightarrow\!\!\!\rightarrow}$.

The following lemma relates the two semantics for service compositions, one expressed in terms of $\longrightarrow$ and the other one in terms of $\hookrightarrow$.

Lemma 3.1 Let Γ be a type environment and C be a service composition such that $\Gamma \rhd C$ is a configuration. It holds that

- $\Gamma \rhd C \overset{\tau}{\hookrightarrow} \Gamma' \rhd C'$ if and only if $C = C_1 \parallel p[\sigma] \parallel C_2$, $C' = C_1 \parallel p[\sigma'] \parallel C_2$ and $\sigma \xrightarrow{\tau} \sigma'$;

- $\Gamma \rhd C \overset{\{a\}_{p\to q}}{\hookrightarrow} \Gamma \rhd C'$ if and only if $C = C_1 \parallel p[\sigma] \parallel C_2 \parallel q[\rho] \parallel C_3$, $C' = C_1 \parallel p[\sigma'] \parallel C_2 \parallel q[\rho'] \parallel C_3$ and $\sigma \xrightarrow{\bar{a}@q} \sigma'$ and $\rho \xrightarrow{a@p} \rho'$.

In order to define our notion of non-interference, we need to be able to distinguish the internal interactions at a given security clearance. As internal transitions are typed, we can assign a security level to them as follows: the level of a synchronization depends on the level of the principals performing it. More precisely, we denote by $\Gamma(\{a\}_{p\to q})$ the level of the internal action $\{a\}_{p\to q}$ in the type environment Γ and we define is as:

$$\Gamma(\{a\}_{p\to q}) = \Gamma(p) \sqcap \Gamma(q).$$

Thus a ς-level synchronization between two components is performed by principals whose security level is higher or equal to ς.

Behavioural observations We define behavioural observations in terms of equivalences that are parametric with respect to the security level $\varsigma \in \Sigma$ of the behaviour we want to observe. Such equivalences are relations over configurations that equate service compositions exhibiting the same ς-level component interactions.

Our behavioural equivalence is a variant of the notion of *weak bisimulation* [16], an observation equivalence which allows one to observe the nondeterministic structure of the LTSs and focuses only on the observable actions.

In the following, we write $\Gamma_1 =_\varsigma \Gamma_2$ whenever $\{p \in dom(\Gamma_1) | \Gamma_1(p) \preceq \varsigma\} = \{p \in dom(\Gamma_2) | \Gamma_2(p) \preceq \varsigma\}$.

Definition 3.2 [Weak bisimulation on ς-low actions] Let $\varsigma \in \Sigma$. A *weak bisimulation on ς-low actions* is the largest symmetric relation $\approx_\varsigma$ over configurations such that whenever $\Gamma_1 \rhd C_1 \approx_\varsigma \Gamma_2 \rhd C_2$ with $\Gamma_1 =_\varsigma \Gamma_2$

- if $\Gamma_1 \rhd C_1 \overset{\alpha}{\hookrightarrow} \Gamma_1' \rhd C_1'$ with $\alpha = \tau$ or $\Gamma_1(\alpha) \preceq \varsigma$, then there exist Γ_2' and C_2' such that $\Gamma_2 \rhd C_2 \overset{\alpha}{\hookrightarrow\!\!\!\rightarrow} \Gamma_2' \rhd C_2'$ with $\Gamma_1' \rhd C_1' \approx_\varsigma \Gamma_2' \rhd C_2'$ and $\Gamma_1' =_\varsigma \Gamma_2'$;

- if $\Gamma_1 \rhd C_1 \overset{\alpha}{\hookrightarrow} \Gamma_1' \rhd C_1'$ with $\alpha \neq \tau$ and $\Gamma(\alpha) \not\preceq \varsigma$, then there exist Γ_2' and C_2' such that

  - either $\Gamma_2 \rhd C_2 \overset{\alpha}{\hookrightarrow\!\!\!\rightarrow} \Gamma_2' \rhd C_2'$
  - or $\Gamma_2 \rhd C_2 \overset{\tau}{\hookrightarrow\!\!\!\rightarrow} \Gamma_2' \rhd C_2'$

  with $\Gamma_1' \rhd C_1' \approx_\varsigma \Gamma_2' \rhd C_2'$ and $\Gamma_1' =_\varsigma \Gamma_2'$.

We write $\Gamma \models C_1 \approx_\varsigma C_2$ whenever $\Gamma \rhd C_1 \approx_\varsigma \Gamma \rhd C_2$. $\square$

The notion of non-interference is inspired by [12] and is expressed in terms of a restriction operator $\cdot|_\varsigma$ which allows one to represent a service composition prevented from performing internal synchronizations of a level higher than ς. The semantics of $C|_\varsigma$ is described by the following rule:

$$\frac{\Gamma \rhd C \overset{\alpha}{\hookrightarrow} \Gamma' \rhd C'}{\Gamma \rhd C|_\varsigma \overset{\alpha}{\hookrightarrow} \Gamma' \rhd C'|_\varsigma}\ \Gamma(\alpha) \preceq \varsigma$$

Table 5 Typed internal composition transitions

Typed internal composition transitions: $\Gamma \rhd C \xhookrightarrow{\alpha} \Gamma' \rhd C'$

$$\frac{\Gamma \rhd \sigma \xrightarrow{\tau} \Gamma' \rhd \sigma'}{\Gamma \rhd p[\sigma] \xhookrightarrow{\tau} \Gamma' \rhd p[\sigma']} \qquad \frac{\Gamma \rhd C_1 \overset{a_{p\to q}}{\longrightarrow} \Gamma \rhd C_1' \quad \Gamma \rhd C_2 \overset{\bar{a}_{p\to q}}{\longrightarrow} \Gamma \rhd C_2'}{\Gamma \rhd C_1 \parallel C_2 \overset{\{a\}_{p\to q}}{\hookrightarrow} \Gamma \rhd C_1' \parallel C_2'} \qquad \frac{\Gamma \rhd C_1 \xhookrightarrow{\alpha} \Gamma' \rhd C_1'}{\Gamma \rhd C_1 \parallel C_2 \xhookrightarrow{\alpha} \Gamma' \rhd C_1' \parallel C_2}$$

Definition 3.3 [Non-interference] Let $\varsigma \in \Sigma$, Γ be a type environment and C be a service composition such that $\Gamma \rhd C$ is a configuration. We say that the service composition C satisfies the non-interference property with respect to the level ς in the type environment Γ, denoted $C \in \mathcal{NI}_{\Gamma,\varsigma}$, if

$$\Gamma \models C \approx_\varsigma C|_\varsigma. \qquad \square$$

Since our calculus for service compositions is finite state, the non-interference property defined above can be checked in polynomial time on the number of states reachable from the initial configuration [12]. In particular, a simple variant of our previous algorithms for the security of CCS processes can be used.

Example 3.4 Consider again the service composition S of Example 2.1 in the type environment Γ of Example 2.3. It holds that $S \in \mathcal{NI}_{\Gamma,\mathtt{L}}$. $\square$

Example 3.5 Consider the service composition depicted in Table 6: it consists of a *client* C, two *financial consulting services* F_1 and F_2 and a *stock quote service provider* S. The client inquires the financial services to get investment advices. The financial services consult the stock quote service provider in order to look up information on the financial quotes. Then the financial services send their investment recommendations to the client which may decide to adhere to the investment plan proposed by one of the financial services and close the connection with the other one.

Let $\Sigma = \{\mathtt{L}, \mathtt{H}\}$ with $\mathtt{L} \preceq \mathtt{H}$ and Γ be the type environment $C : \mathtt{H}$, $F_1 : \mathtt{L}$, $F_2 : \mathtt{L}$, $S : \mathtt{L}$. In this case we have that $M \notin \mathcal{NI}_{\Gamma,\mathtt{L}}$. Indeed, there is a direct causality between the high level actions $\{\mathtt{Agree}\}_{C\to F_i}$ and the low level action $\{\mathtt{Close}\}_{C\to F_j}$ with $i \neq j$, performed after the clients makes the choice. As a consequence, if the client decides to accept the proposal of F_1 then F_2 knows that the customer has agreed to proceed with investment recommendation of F_1 by just observing that the action $\{\mathtt{Close}\}_{C\to F_2}$ has been performed. The service composition can be made secure by letting $\{\mathtt{Close}\}_{C\to F_j}$ be executed independently from $\{\mathtt{Agree}\}_{C\to F_i}$ as in the composition M' which is obtained from M by replacing the contract σ_C with σ_C'. $\square$

4 Compliance and Non-Interference

Compliance is a basic property that characterizes the correct behavior of concurrent distributed systems. It is used widely in the context of Service Oriented Architectures as a formal device to identify well behaving service compositions, those whose interactions are free of synchronization errors.

In this paper we refer to the notion of compliance for service contract compositions studied in [2]. Intuitively, a composition of services is compliant if it is deadlock and livelock free, i.e., it does not get stuck nor does it get trapped into infinite loops with no exit states. This notion is independent from the security levels of principals involved in the internal synchronizations. Thus we omit trailing type environments in the definitions below, and write, e.g., $C \Longrightarrow C'$ to denote a transition of the form $\Gamma \rhd C \Longrightarrow \Gamma' \rhd C'$ for some type environments Γ and Γ'.

Definition 4.1 [Compliance] Let C be a service composition. We say that C is *compliant*, noted $C\downarrow$, if for every C' such that $C \Longrightarrow C'$ there exists C'' such that $C' \Longrightarrow C''$ and $C''\checkmark$. $\square$

Based on the notion of compliance, one can introduce a compliance pre-order on the component contracts in terms of their ability to preserve the compliance of the service compositions they are part of. This is formalized through the notion of *subcontract relation*. Intuitively, compositions are seen as tests for comparing single contracts. Two contracts are related if so are the sets of service compositions compliant with them.

Formally, we first define the set of compositions compliant with a contract σ relative to a principal p as: $[\sigma]_p = \{C | \ (C \parallel p[\sigma])\downarrow\}$. Then we express the subcontract relation in terms of set inclusion as follows.

Definition 4.2 [Subcontract Relation] Let p be a principal and σ and ρ be two p-compatible contracts. We say that σ is a subcontract for ρ relative to the principal p, written $\sigma \sqsubseteq_p \rho$, if $[\sigma]_p \subseteq [\rho]_p$. $\square$

The subcontract relation does not in general preserve non-interference.

Table 6 Example of a financial consulting platform

$$M = C[\sigma_C] \parallel F_1[\sigma_F] \parallel F_2[\sigma_F] \parallel S[\sigma_S]$$

$$\sigma_C = \overline{\mathtt{Inq}}@F_1.\overline{\mathtt{Inq}}@F_2.\mathtt{Plan}@F_1.\mathtt{Plan}@F_2.(\,\overline{\mathtt{Agree}}@F_1.\overline{\mathtt{Close}}@F_2.\mathbf{1}\lfloor_{F_1:\mathtt{H}}\oplus\,_{F_2:\mathtt{H}}\rfloor\overline{\mathtt{Agree}}@F_2.\overline{\mathtt{Close}}@F_1.\mathbf{1}\,)\,)$$

$$\sigma_F = \mathtt{Inq}@x.\overline{\mathtt{LookUp}}@S.\mathtt{Quote}@x.\overline{\mathtt{Plan}}@C.(\,\mathtt{Agree}@x.\mathbf{1} + \mathtt{Close}@x.\mathbf{1}\,)$$

$$\sigma_S = \mathtt{LookUp}@x.\overline{\mathtt{Quote}}@x.\mathbf{1}$$

$$M' = C[\sigma'_C] \parallel F_1[\sigma_F] \parallel F_2[\sigma_F] \parallel S[\sigma_S]$$

$$\sigma'_C = \overline{\mathtt{Inq}}@F_1.\overline{\mathtt{Inq}}@F_2.\mathtt{Plan}@F_1.\mathtt{Plan}@F_2.(\,\overline{\mathtt{Close}}@F_2.\overline{\mathtt{Agree}}@F_1.\mathbf{1}\lfloor_{F_1:\mathtt{H}}\oplus\,_{F_2:\mathtt{H}}\rfloor\overline{\mathtt{Close}}@F_1.\overline{\mathtt{Agree}}@F_2.\mathbf{1}\,)\,)$$

Table 7 Example of a travel agency (revised)

$$S' = C[\sigma'_C] \parallel T[\sigma_T] \parallel A_1[\sigma_A] \parallel A_2[\sigma_A]$$

$$\sigma'_C = \overline{\mathtt{Req}}@T.\mathtt{Lst}@T.(\,\overline{\mathtt{Close}}@T.\mathbf{1}\lfloor_{\emptyset}\oplus\,_{T:\mathtt{H}}\rfloor\overline{\mathtt{Buy1}}@T.\overline{\mathtt{Pay}}@T.\mathtt{Get}@A_1.\mathbf{1}\,)$$

Example 4.3 Consider the service composition S' obtained from S by replacing the contract σ_C with σ'_C as in Table 7. We have that $\sigma_C \sqsubseteq_C \sigma'_C$ but $S \in \mathcal{NI}_{\Gamma,\mathtt{L}}$ does not imply $S' \in \mathcal{NI}_{\Gamma,\mathtt{L}}$ since the service has a branch that performs two high actions with a final low action. $\square$

In [7] we introduced a safe coinductive preorder for this subcontract relation. It provides an efficient method to check whether a service component can be safely replaced inside a multi-party service composition. Here we show that, under some conditions, the same verification techniques can be used to check whether a contract can be replaced inside a composition while preserving the non-interference property with respect to a given security level ς in a specific type environment Γ. In particular, the next theorem states that if $\sigma \sqsubseteq_p \rho$ then we can replace the component $p[\sigma]$ with $p[\rho]$ preserving the non-interference property with respect to a security level ς in a type environment Γ provided that p has level lower or equal to ς in Γ and the level of p will never be upgraded above ς.

We generalize the definition of $\coprod$ in Table 3 to service components as follows: $\coprod(p_1[\sigma_1] \parallel \cdots \parallel p_n[\sigma_n]) = \coprod(\sigma_1) \sqcup \cdots \coprod(\sigma_n)$.

Theorem 4.4 Let p be a principal and σ and ρ be two p-compatible contracts. If $\sigma \sqsubseteq_p \rho$ then for all type environment Γ and security level ς such that $\Gamma(p) \preceq \varsigma$ then

- $(C \parallel p[\sigma]) \in \mathcal{NI}_{\Gamma,\varsigma}$ implies $(C \parallel p[\rho]) \in \mathcal{NI}_{\Gamma,\varsigma}$ for every C with $\coprod(C)(p) \preceq \varsigma$.

A more general result can be obtained by defining a stronger coinductive subcontract preorder with respect to the one in [7] which preserves both compliance and non-interference. First we need some notation.

Let $\mathtt{R} \subseteq \{\bar{a}@p, a@p \mid a \in \mathcal{A},\ p \in \mathcal{P}\} \cup \{\,\checkmark\}$ be a set of contract actions which may contain both visible actions and the satisfaction signal $\checkmark$. We write $\sigma \downarrow \mathtt{R}$ to note that $\mathtt{R}$ is the smallest non-empty set such that $\sigma \xrightarrow{\lambda}$ with $\lambda \neq \tau$ implies $\lambda \in \mathtt{R}$, and $\sigma\checkmark$ implies $\checkmark \in \mathtt{R}$. Moreover, we write $\sigma \Downarrow \mathtt{R}$ whenever $\sigma \Longrightarrow \sigma'$ with $\sigma' \downarrow \mathtt{R}$.

The next definition provides a safe coinductive proof method to check whether a contract σ can be replaced by ρ in a service composition while preserving both compliance and non-interference.

Definition 4.5 [Strong Coinductive Subcontract Relation] A strong coinductive subcontract relation is the largest relation $\preceq$ over contracts such that whenever $\sigma \preceq \rho$

- if $\rho \xrightarrow{\tau} \rho'$ then $\sigma \preceq \rho'$,
- if $\rho \downarrow \mathtt{R}$ then for every $\mathtt{S}$ such that $\sigma \Downarrow \mathtt{S}$ it holds $\mathtt{S} \subseteq \mathtt{R}$,
- if $\rho \xrightarrow{\lambda} \rho'$ with $\lambda \neq \tau$ then $\sigma \Longrightarrow \sigma' \xrightarrow{\lambda} \sigma''$ and $\sigma'' \preceq \rho'$. $\square$

Theorem 4.6 Let p be a principal and σ and ρ be two p-compatible contracts. If $\sigma \preceq \rho$ then for all principal p, type environment Γ and security level ς it holds that

- $(C \parallel p[\sigma])\downarrow$ implies $(C \parallel p[\rho])\downarrow$, and
- $(C \parallel p[\sigma]) \in \mathcal{NI}_{\Gamma,\varsigma}$ implies $(C \parallel p[\rho]) \in \mathcal{NI}_{\Gamma,\varsigma}$.

5 Conclusion

We have developed a formal framework for the analysis of non-inteference of service compositions. We consider a simple typing discipline where types are used to assign security levels to service components. We allow types to be dynamically specified by the service participants. Our framework is effective: efficient algorithms for (weak) bisimulation can be adapted (see, e.g., [17]) in order to check the non-interference property in polynomial time on the number of states reachable from the initial configuration.

Future work concerns the development of a unique framework for a formal analysis of service non-interference, compliance, conformance, adaption and replacement inside a composition.

References

[1] Anne H. Anderson. An introduction to the web services policy language (wspl). In *5th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2004)*, pages 189–192. IEEE Computer Society, 2004.

[2] G. Bernardi, M. Bugliesi, D. Macedonio, and S. Rossi. A theory of adaptable contract-based service composition. In *GlobalComp'08*. IEEE Computer Society, 2008.

[3] Karthikeyan Bhargavan, Cédric Fournet, and Andrew D. Gordon. Verifying policy-based web services security. *ACM Trans. Program. Lang. Syst.*, 30(6):1–59, 2008.

[4] M. Bravetti and G. Zavattaro. Contract based multi-party service composition. In *FSEN'07*, volume 4767 of *LNCS*, pages 207–222. Springer, 2007.

[5] M. Bravetti and G. Zavattaro. Towards a unifying theory for choreography conformance and contract compliance. In *SC'07*, volume 4829 of *LNCS*, pages 34–50. Springer, 2007.

[6] M. Bravetti and G. Zavattaro. Contract compliance and choreography conformance in the presence of message queues. In *WS-FM'08*, volume 5387 of *LNCS*, pages 37–54. Springer, 2008.

[7] M. Bugliesi, D. Macedonio, L. Pino, and S. Rossi. Compliance-based testing theories for web services. Technical Report CS-2009-5, Dipartimento di Informatica, Università Ca' Foscari, Venezia, 2009.

[8] S. Carpineti, G. Castagna, C. Laneve, and L. Padovani. A formal account of contracts for web services. In *WS-FM'06*, volume 4184 of *LNCS*, page 148162. Springer-Verlag, 2006.

[9] G. Castagna, N. Gesbert, and L. Padovani. A theory of contracts for web services. In *POPL'08*, pages 261–272. ACM press, 2008.

[10] G. Castagna, N. Gesbert, and L. Padovani. A theory of contracts for web services. *ACM Transactions on Programming Languages and Systems*, 2009. To appear.

[11] R. Focardi and R. Gorrieri. A Classification of Security Properties for Process Algebras. *Journal of Computer Security*, 3(1):5–33, 1994/1995.

[12] R. Focardi and S. Rossi. Information Flow Security in Dynamic Contexts. *Journal of Computer Security*, 14(1):65–110, 2006.

[13] J. A. Goguen and J. Meseguer. Security Policies and Security Models. In *Proc. of the IEEE Symposium on Security and Privacy (SSP'82)*, pages 11–20. IEEE Computer Society Press, 1982.

[14] Cosimo Laneve and Luca Padovani. The *must* preorder revisited. In *CONCUR*, pages 212–225, 2007.

[15] J. McLean. Security Models and Information Flow. In *Proc. of the IEEE Symposium on Security and Privacy (SSP'90)*, pages 180–187. IEEE Computer Society Press, 1990.

[16] R. Milner. *Communication and Concurrency*, volume 92 of *Prentice Hall International Series in Computer Science*. Prentice Hall, 1989.

[17] C. Piazza, E. Pivato, and S. Rossi. CoPS: Checker of Persistent Security. In *Proc. of Int. Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS'04)*, volume 2988 of *LNCS*, pages 144–152. Springer, 2004.

[18] P.Y.A. Ryan and S. Schneider. Process Algebra and Non-Interference. *Journal of Computer Security*, 9(1/2):75–103, 2001.

[19] A. Sabelfeld and A. C. Myers. Language-Based Information-Flow Security. *IEEE Journal on Selected Areas in Communication*, 21(1):5–19, 2003.

[20] A. Zakinthinos and E. S. Lee. A General Theory of Security Properties. In *Proc. of the IEEE Symposium on Security and Privacy (SSP'97)*, pages 74–102. IEEE Computer Society Press, 1997.