Loosening the notions of compliance and sub-behaviour in client/server systems

Franco Barbanera
Dipartimento di Matematica e Informatica
University of Catania
barba@dmi.unict.it

Ugo de'Liguoro
Dipartimento di Informatica
University of Torino
ugo.deliguoro@di.unito.it

In the context of "session behaviors" for client/server systems, we propose a weakening of the compliance and sub-behaviour relations where the bias toward the client (whose "requests" must be satisfied) is pushed further with respect to the usual definitions, by admitting that "not needed" output actions from the server side can be *skipped* by the client. Both compliance and sub-behaviour relations resulting from this weakening remain decidable, though the proof of the duals-as-minima property for servers, on which the decidability of the sub-behaviour relation relies, requires a tighter analysis of client/server interactions.

1 Introduction

The formal specification of web-services behaviour is a crucial issue toward automatic discovery and composition of software modules available through a network. Among several approaches we consider here the theory of contracts introduced in [6] and developed in a series of papers e.g. [8, 7]. We focus here on the scenario of client/server architecture, where services stored in a repository are queried by clients to establish a two-sided communication.

To check the matching of client's requirements against the service offered by a server, both server and client behaviours are described via a CCS-like formalism (without τ-actions nor parallel composition), whose terms are dubbed *contracts*. The basic notion studied in the theory is the *compliance* relation, written $\rho \dashv \sigma$, meaning that all requirements by the client ρ are eventually matched by some communication action by the server σ [1]. This is mathematically defined using an LTS semantics of the communication behaviour of the pair of contracts $\rho\|\sigma$, where $\rho\|\sigma \longrightarrow \rho'\|\sigma'$ holds whenever $\rho \xrightarrow{\alpha} \rho'$, $\sigma \xrightarrow{\overline{\alpha}} \sigma'$ and α and $\overline{\alpha}$ are dual actions. Now, writing $\Longrightarrow$ for the reflexive and transitive closure of $\longrightarrow$, the relation $\rho \dashv \sigma$ holds if and only if $\rho\|\sigma \Longrightarrow \rho'\|\sigma' \not\Longrightarrow$ implies $\rho' = \mathbf{1}$, where $\mathbf{1}$ is the behaviour of the completed process. When $\rho \dashv \sigma$ we say that ρ is a *client* of the *server* σ, slightly abusing terminology.

The compliance relation characterises client/server interaction with a bias toward the client, which is the sole guaranteed to complete. To illustrate this by an example, let us consider a ballot service whose behaviour is described by the following server contract:

$$\mathsf{BallotServiceAB} \triangleq \mathsf{rec}\,x.\ \mathtt{Login}.(\overline{\mathtt{Wrong}}.x \oplus \overline{\mathtt{Overload}}.x \oplus \overline{\mathtt{Ok}}.(\mathtt{VoteA} + \mathtt{VoteB})).$$

This service can receive a login from a client, a voter, via the input action `Login`; if the login is correct the server issues to the client the message $\overline{\mathtt{Ok}}$ (an output action), enabling the client to vote for either candidates A or B via a continuation consisting of the external choice $+$ of the input actions `VoteA` and

[1] It is not feasible, however, to allow the client to terminate the interaction at any point, since, trivially, any server would be compliant with such a sort of client.

I. Lanese, A. Lluch Lafuente, A. Sokolova, H. T. Vieira (Eds.):
7th Interaction and Concurrency Experience (ICE 2014)
EPTCS 166, 2014, pp. 94–110, doi:10.4204/EPTCS.166.10

VoteB. In case the login is incorrect or the service is busy, the messages $\overline{\texttt{Wrong}}$ and $\overline{\texttt{Overload}}$ are sent to the client respectively, both by output actions. In both cases the voter is allowed to retry the login by recursion. The output actions $\overline{\texttt{Ok}}$, $\overline{\texttt{Wrong}}$ and $\overline{\texttt{Overload}}$ are composed by an internal choice $\oplus$ since they depend on internal decisions on the server side. Now let us consider the following client:

$$\mathsf{Voter} \triangleq \mathsf{rec}\,x.\ \overline{\texttt{Login}}.(\texttt{Wrong}.x + \texttt{Overload}.x + \texttt{Ok}.\overline{\texttt{VoteA}}).$$

Voter will not give up synchronizing with BallotService until eventually allowed to send her vote. According to the definition of compliance we have that Voter $\dashv$ BallotServiceAB, and this remains true also in the case of the slightly different server:

$$\mathsf{BallotServiceABC} \triangleq \mathsf{rec}\,x.\ \texttt{Login}.(\overline{\texttt{Wrong}}.x \ \oplus\ \overline{\texttt{Ok}}.(\texttt{VoteA} + \texttt{VoteB} + \texttt{VoteC})).$$

which is not willing to issue the $\overline{\texttt{Overload}}$ message, and allows one more candidate to be voted. Indeed what matters is the fact that no interaction among client and server will ever get stuck in a state in which some client action is pending. Because of the same reason the client Voter is also compliant with the service:

$$\mathsf{rec}\,x.\ \texttt{Login}.(\overline{\texttt{Wrong}}.x \ \oplus\ \overline{\texttt{Overload}}.x \ \oplus\ \overline{\texttt{Ok}}.(\texttt{VoteA}.(\texttt{Va1} + \texttt{Va2}) + \texttt{VoteB}.(\texttt{Vb1} + \texttt{Vb2}))),$$

where Va1 and Va2 are choices depending on the vote VoteA, and similarly for Vb1 and Vb2. However Voter is not compliant with

$$\begin{array}{l}\mathsf{BallotServiceBehSkp} \triangleq \\ \qquad \mathsf{rec}\,x.\ \texttt{Login}.(\overline{\texttt{Wrong}}.\overline{\texttt{InfoW}}.x \ \oplus\ \overline{\texttt{Overload}}.x \ \oplus\ \overline{\texttt{Ok}}.\overline{\texttt{Id}}.(\ \ \texttt{VoteA}.(\texttt{Va1} + \texttt{Va2}) \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad + \texttt{VoteB}.(\texttt{Vb1} + \texttt{Vb2})\ )\ )\end{array}$$

because of the actions $\overline{\texttt{InfoW}}$ and $\overline{\texttt{Id}}$ (the former representing infos about the failure of the login and the latter representing an identifier of the voting transaction), that do not have any correspondent input on the client side. However these outputs have hardly any control significance, which is especially the case in the setting of *session-behaviours* we have introduced in [2] and that have been also investigated in [5] (where they are dubbed *session contracts*). In fact session-behaviours are contracts in which the only terms that can occur in an internal choice have to be prefixed by the output of pairwise distinct messages (the internal choice being the only truly non-deterministic feature of a session-behaviour).

In this paper we investigate the possibility of loosening the notion of compliance for session behaviours by admitting that a client, before an actual syncronization, can *skip* (disregard) a finite number of consecutive output actions by the server, provided that these are not the dual of some immediate input actions of the client. The overall number of (non consecutive) skipped actions in an interaction, however, can be possibly infinite. We call the resulting relation skp-*compliance* and write $\rho \dashv^{\mathsf{skp}} \sigma$ for "ρ is skp-compliant with σ". There is a contrast between these two conditions; while the latter is easily decidable by looking at the contract syntax (and by admitting only guarded recursion), the former is an infinitary condition, ruling out those infinite interactions which happen to be *definitely* skip actions. The first result which we obtain is that, in spite of its infinitary definition, the so obtained compliance notion is decidable.

Compliance naturally induces a preorder over contracts seen as the behavioural specification of a server. In [2, 4] we say that $\sigma \preceq_s \sigma'$ if any client of σ is also a client of σ' according to the compliance relation $\dashv$. It can be checked that, for example, BallotServiceAB $\preceq_s$ BallotServiceABC, but neither of them is comparable to BallotServiceBehSkp. By replacing $\dashv^{\mathsf{skp}}$ in this definition one obtains a similar preorder $\sigma \preceq^{\mathtt{skp}} \sigma'$, which also turns out to be decidable. The proof of the latter fact relies on the notion of dual behaviour $\overline{\rho}$ of ρ and on the property that $\overline{\rho}$ is the minimal server of ρ w.r.t. $\preceq^{\mathtt{skp}}$.

Overview of the paper. The notion of session-behaviour is recalled in Section 2. Then the definition of skp-compliance is given in Subsection 2.1. In Section 3 it is provided a coinductive characterization of skp-compliance, via a formal system to deduce (conditional) skp-compliance, which is proved to be sound and complete. Decidability then follows, being the system algorithmic. The notion of skp-subbehaviour $\preceq^{\mathtt{skp}}$ is introduced in Section 4, and the property of *duals as minima* is proved. Decidability of $\preceq^{\mathtt{skp}}$ is a consequence of such a property. In Section 5 we extensively discuss the relationship of our skp-compliance with another weak notion of compliance allowing for a sort of "action skipping": the (orchestrated) weak-compliance proposed by Padovani in [14]. In Section 6 a discussion about future works concludes the paper.

2 Session Behaviours and the skp-compliance relation

Contracts [12, 9] are a subset of CCS terms, defined by the grammar :

$$\sigma ::= \mathbf{1} \mid \alpha.\sigma \mid \sigma + \sigma \mid \sigma \oplus \sigma \mid x \mid \mathsf{rec}\, x.\sigma$$

where α ranges over a set of actions and co-actions, $\mathbf{1}$ is the same as the CCS term 0, namely the completed protocol, $+$ and $\oplus$ are external and internal choices respectively. *Session behaviours* as defined below are a further restriction of this set. They are designed to be in one-to-one correspondence to session types [11] without delegation (in [2] and [4] session behaviours were extended by send/receive actions of session behaviours to model delegation). The restriction is achieved by constraining internal and external choices in a way that limits the non-determinism to (internal) output selection.

Definition 2.1 (Session Behaviours)

i) *Let $\mathcal{N}$ be some countable set of symbols and $\overline{\mathcal{N}} = \{\overline{a} \mid a \in \mathcal{N}\}$, with $\mathcal{N} \cap \overline{\mathcal{N}} = \emptyset$.*
The set BE *of* raw behaviour expressions *is defined by the following grammar:*

$$\begin{array}{rcll} \sigma, \tau & ::= & \mathbf{1} & \textit{inaction} \\ & & \mid\; a_1.\sigma_1 + \cdots + a_n.\sigma_n & \textit{external choice} \\ & & \mid\; \overline{a}_1.\sigma_1 \oplus \cdots \oplus \overline{a}_n.\sigma_n & \textit{internal choice} \\ & & \mid\; x & \textit{variable} \\ & & \mid\; \mathsf{rec}\, x.\sigma & \textit{recursion} \end{array}$$

where

- $n \geq 1$ *and* $a_i \in \mathcal{N}$ *(hence* $\overline{a}_i \in \overline{\mathcal{N}}$*) for all* $1 \leq i \leq n$;
- x *is a session behaviour variable out of a denumerable set and it is bound by the* rec *operator.*

As usual, σ is said to be closed *whenever* $\mathrm{FV}(\sigma) = \emptyset$, *where* $\mathrm{FV}(\sigma)$ *denotes the set of free variables in σ.*

ii) *The set* SB *of* **session behaviours** *is the subset of closed raw behaviour expressions such that in $a_1.\sigma_1 + \cdots + a_n.\sigma_n$ and $\overline{a}_1.\sigma_1 \oplus \cdots \oplus \overline{a}_n.\sigma_n$, the a_i and the $\overline{a}_i$ are, respectively, pairwise distinct; moreover in $\mathsf{rec}\, x.\sigma$ the expression σ is not a variable.*

We abbreviate $a_1.\sigma_1 + \cdots + a_n.\sigma_n$ by $\sum_{i=1}^{n} a_i.\sigma_i$, and $\overline{a}_1.\sigma_1 \oplus \cdots \oplus \overline{a}_n.\sigma_n$ by $\bigoplus_{i=1}^{n} \overline{a}_i.\sigma_i$. We also use the notations $\sum_{i\in I} a_i.\sigma_i$ and $\bigoplus_{i\in I} \overline{a}_i.\sigma_i$, for finite and not empty I. The trailing $\mathbf{1}$ is normally omitted: we write e.g. $a+b$ for $a.\mathbf{1}+b.\mathbf{1}$.

Note that recursion in SB is guarded and hence contractive in the usual sense [1]. Session behaviours will be considered modulo commutativity of internal and external choices.

A syntactical notion of *duality* on SB is easily obtained by interchanging a with $\overline{a}$, and $+$ with $\oplus$. Its formal definition can obtained by restricting to SB a straightforward definition by induction on the structure of the raw expressions in BE (i.e. also for open expressions[2]). The dual of a session-behaviour σ will be denoted, as usual, by $\overline{\sigma}$. As expected, $\overline{\overline{\sigma}} = \sigma$ for all σ.

The operational semantics of session behaviours is given in terms of a labeled transition system (LTS) $\sigma \xrightarrow{\alpha} \sigma'$ where $\sigma,\sigma' \in \mathsf{SB}$ and α belongs to an appropriate set of actions $\mathbf{Act}$.

Definition 2.2 (Behaviour LTS)
Let $\mathtt{skp} \notin \mathscr{N}$ *and define the set of actions* $\mathbf{Act} = \mathscr{N} \cup \overline{\mathscr{N}}$ *and* $\oplus, \mathsf{rec} \notin \mathbf{Act}$*; then define the LTS* $(\mathsf{SB}, \mathbf{Act} \cup \{\oplus, \mathsf{rec}\}, \longrightarrow)$ *by the rules:*

$$a_1.\sigma_1 + \cdots + a_n.\sigma_n \xrightarrow{a_k} \sigma_k \qquad\qquad \overline{a}.\sigma \xrightarrow{\overline{a}} \sigma$$

$$\overline{a}_1.\sigma_1 \oplus \cdots \oplus \overline{a}_n.\sigma_n \xrightarrow{\oplus} \overline{a}_k.\sigma_k \qquad\qquad \mathsf{rec}\,x.\sigma \xrightarrow{\mathsf{rec}} \sigma\{\mathsf{rec}\,x.\sigma/x\}$$

where $1 \le k \le n$ *and* $\sigma \xrightarrow{\alpha} \gamma$ *abbreviates* $(\sigma,\alpha,\gamma) \in \longrightarrow$.

We abbreviate $\longrightarrow = \xrightarrow{\oplus} \cup \xrightarrow{\mathsf{rec}}$. Note that neither $\oplus$ nor rec are actions, so that they are unobservable and used just for technical reasons; indeed we adopt the standard $\longrightarrow$ (from CCS without τ) in the subsequent definition of the parallel operator for testing. As usual, we write $\Longrightarrow = \longrightarrow^*$ and $\stackrel{\alpha}{\Longrightarrow} = \longrightarrow^* \xrightarrow{\alpha} \longrightarrow^*$ for $\alpha \in \mathbf{Act}$.

We observe that if $\sigma \stackrel{\alpha}{\Longrightarrow} \sigma'$ or $\sigma \Longrightarrow \sigma'$ for $\sigma \in \mathsf{SB}$, then $\sigma' \in \mathsf{SB}$.

Lemma 2.3 *For any* $\sigma \in \mathsf{SB}$ *there exists a unique and finite set* $R = \{\sigma' \in \mathsf{SB} \mid \sigma \Longrightarrow \sigma' \not\Longrightarrow\}$, *which is either of shape* $\{\mathbf{1}\}$ *or* $\{a_1.\sigma_1 + \cdots + a_n.\sigma_n\}$ *or* $\{\overline{a}_i.\sigma_i \mid i \in I\}$. *Moreover* R *is computable in* σ.

Proof. By induction of the structure of σ. Since recursion is guarded and internal choices are finitary, no infinite $\longrightarrow$ reductions are possible out of σ; on the other hand if $\sigma \in \mathsf{SB}$ then it is closed, so the case $\sigma \Longrightarrow x$ for some variable x is impossible. □

In the sequel we write $\sigma \Downarrow \mathbf{1}$ and $\sigma \Downarrow \sum_{i\in I} a_i.\rho_i$ if the R in the above lemma is, respectively, of the first two shapes, and write $\sigma \Downarrow \bigoplus_{i\in I} \overline{a}_i.\sigma_i$ if $R = \{\overline{a}_i.\sigma_i \mid i \in I\}$.

We shall denote finite or infinite sequences of elements of $\mathbf{Act}$, i.e. elements of $\mathbf{Act}^* \cup \mathbf{Act}^\infty$, by bold characters $\boldsymbol{\alpha}, \boldsymbol{\beta}, \ldots$. Bold italic (overlined) characters $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \ldots$ ($\overline{\boldsymbol{a}}, \overline{\boldsymbol{b}}, \overline{\boldsymbol{c}}, \ldots$) shall denote sequences of elements of $\mathscr{N}$ (resp. $\overline{\mathscr{N}}$). We shall represent the fact that a sequence $\boldsymbol{\alpha}$ is infinite by writing $\boldsymbol{\alpha}^\infty$. The length of a sequence $\boldsymbol{\alpha}$ will be denoted by $|\boldsymbol{\alpha}|$, and it is either finite or ∞.

We write $\sigma \stackrel{\boldsymbol{\alpha}}{\Longrightarrow} \sigma'$ if $\boldsymbol{\alpha} = \alpha_1 \cdots \alpha_n$ and $\sigma \stackrel{\alpha_1}{\Longrightarrow} \cdots \stackrel{\alpha_n}{\Longrightarrow} \sigma'$. Also we write $\sigma \longrightarrow$ and $\sigma \xrightarrow{\alpha}$ if there exists σ' s.t. $\sigma \longrightarrow \sigma'$ and $\sigma \xrightarrow{\alpha} \sigma'$ respectively, and $\sigma \not\longrightarrow$ when $\neg(\sigma \longrightarrow)$. Given $\boldsymbol{\alpha} = \alpha_1 \ldots \alpha_n$ the notation $\beta \in \boldsymbol{\alpha}$ will stand for $\beta \in \{\alpha_1, \ldots, \alpha_n\}$.

We define the set of traces of a session behaviour as follows.

[2] To avoid too cumbersome definitions, any time an inductive definition on elements of SB will be provided, it will be tacitly assumed to be actually the restriction to SB of the corresponding inductive definition on BE.

Definition 2.4 (Traces) *The mapping* $\mathsf{Tr} : \mathsf{SB} \to (\mathscr{P}(\mathbf{Act}^*) \cup \mathscr{P}(\mathbf{Act}^\infty))$ *is defined by*

$$\mathsf{Tr}(\textstyle\sum_{i\in I} a_i.\sigma_i) = \bigcup_{i\in I}\{a_i\,\boldsymbol{\alpha} \mid \boldsymbol{\alpha} \in \mathsf{Tr}(\sigma_i)\} \qquad \mathsf{Tr}(\mathbf{1}) = \{\varepsilon\}$$
$$\mathsf{Tr}(\textstyle\bigoplus_{i\in I} \overline{a}_i.\sigma_i) = \bigcup_{i\in I}\{\overline{a}_i\,\boldsymbol{\alpha} \mid \boldsymbol{\alpha} \in \mathsf{Tr}(\sigma_i)\} \qquad \mathsf{Tr}(\mathsf{rec}\,x.\sigma) = \mathsf{Tr}(\sigma[\mathsf{rec}\,x.\sigma/x])$$

A session-behaviours σ is said to be *finite* whenever $\mathsf{Tr}(\sigma) \in \mathscr{P}(\mathbf{Act}^*)$.

2.1 The skp-compliance relation

As for contract compliance, we use an LTS of client/server pairs $\rho\|\sigma$ to define the notion of skp-compliance on session-behaviours. The actions of the LTS are the silent action τ, representing a full handshake between synchronizing actions on the client and server sides, together with a "skipping" action skp, representing the fact that an action on the server's side has been discarded.

As mentioned in the introduction, we allow only output actions on the server side to be discarded. However we disallow the skip of an output action that synchronizes with some input action by the client. Let us write:

$$\rho \not\Downarrow \alpha \Leftrightarrow \neg\exists \rho'.\, \rho \stackrel{\alpha}{\Longrightarrow} \rho'.$$

Observe that the statement $\rho \not\Downarrow \alpha$ is decidable because it is the negation of $\sigma \Downarrow \sum_{i\in i} a_i.\sigma_i$ or $\sigma \Downarrow \bigoplus_{i\in i} \overline{a}_i.\sigma_i$, with $\alpha \in \{a_i, \overline{a}_i \mid i \in I\}$, which are decidable by Lemma 2.3.

The next definitions formally introduce the LTS for client/server pairs and the relation of skp-*compliance* for session behaviours, that we dub $\dashv^{\mathtt{skp}}$.

Definition 2.5 (LTS for Client-Server pairs)
Let $\mathbf{sAct} = \{\tau, \mathtt{skp}\}$ *be the set of the synchronization actions and* $\rho\|\sigma$ *denote the parallel composition of session behaviors in* SB*, then define:*

$$\frac{\rho \longrightarrow \rho'}{\rho\|\sigma \longrightarrow \rho'\|\sigma} \qquad \frac{\sigma \longrightarrow \sigma'}{\rho\|\sigma \longrightarrow \rho\|\sigma'}$$

$$\frac{\rho \not\Downarrow a \quad \sigma \stackrel{\overline{a}}{\longrightarrow} \sigma'}{\rho\|\sigma \stackrel{\mathtt{skp}}{\longrightarrow} \rho\|\sigma'} \qquad \frac{\rho \stackrel{\alpha}{\longrightarrow} \rho' \quad \sigma \stackrel{\overline{\alpha}}{\longrightarrow} \sigma'}{\rho\|\sigma \stackrel{\tau}{\longrightarrow} \rho'\|\sigma'}$$

where $a \in \mathscr{N}$ *(and hence* $\overline{a} \in \overline{\mathscr{N}}$*),* $\alpha \in \mathbf{Act}$ *and* $\overline{\alpha}$ *is its dual, such that* $\overline{\overline{\alpha}} = \alpha$*.*

The ratio of introducing the ability of clients to skip some actions on the server side is to allow more clients to synchronize with servers that essentially provide the required service but for some supplementary (and possibly redundant) information.

We abbreviate $\Longrightarrow \;=\; \longrightarrow^*$ and $\stackrel{\xi}{\Longrightarrow} \;=\; \Longrightarrow \circ \stackrel{\xi}{\longrightarrow} \circ \Longrightarrow$, where $\xi \in \mathbf{sAct}$. Moreover, by $\stackrel{\zeta^*\xi}{\Longrightarrow}$ we denote $\stackrel{\zeta}{\Longrightarrow}^* \circ \stackrel{\xi}{\Longrightarrow}$, where $\zeta, \xi \in \mathbf{sAct}$.

Remark 2.6 We observe that it would be unreasonable to allow clients to deny replies to server input actions, as this would result into a complete loss of control (think of the Login action in the ballot service examples). On the other hand we balance the possibility of skipping server outputs by two principles.

The first one is that the client is not allowed to defer the synchronization with an output action of the server which it is ready to accept, avoiding the indeterminacy of synchronizations like

$$a\|\overline{b}.\overline{a}.\overline{b}.\overline{a} \xrightarrow{\texttt{skp}} a\|\overline{a}.\overline{b}.\overline{a} \xrightarrow{\tau} \mathbf{1}\|\overline{b}.\overline{a}$$

and

$$a\|\overline{b}.\overline{a}.\overline{b}.\overline{a} \xrightarrow{\texttt{skp}} a\|\overline{a}.\overline{b}.\overline{a} \xrightarrow{\texttt{skp}} a\|\overline{b}.\overline{a} \xrightarrow{\texttt{skp}} a\|\overline{a} \xrightarrow{\tau} \mathbf{1}\|\mathbf{1}$$

of which only the first one is legal. The second principle is that a client has not to be compliant with a server that will never provide the required output. This happens in an infinite interaction which is *definitely* made of skp-synchronization actions, as in the cases of $\mathsf{rec}\,x.b.x\|\mathsf{rec}\,x.\overline{a}.x$ and of the subtler $b\|\mathsf{rec}\,x.(\overline{a}.x \oplus \overline{b})$.
However, it is reasonable to allow the overall number of skippings to be infinite. A simple example of that is when all the infinite b's of the client $\mathsf{rec}\,x.b.x$ manage to syncronize with a $\overline{b}$ of the server $\mathsf{rec}\,x.\overline{a}.\overline{a}.\overline{b}.x$, each time skipping the $\overline{a}$ preceding the $\overline{b}$ and the $\overline{a}$ following it.

So, as previously discussed, the notion of compliance we wish to formalize is an extension of the usual notion of compliance such that any finite or infinite number of output actions from the server can be discarded. We wish however to rule out the possibility of a client indefinitely discarding output actions coming from the server. So, in order to do that, we formalize below the skp-compliance relation in terms of *synchronization traces*. A synchronization trace describes a possible client/server interaction as a sequence of successful handshakes (τ) or skipping actions (skp). Such traces can be either finite or infinite. A client will then be compliant with a server when all the client/server finite synchronization traces ends with ✓ (which can occur only in case the client completes, i.e. gets to **1**) and all the infinite synchronization traces are not formed of just skp elements from a certain point on, i.e. are not *definitely*-skp.

Definition 2.7 (Synchronization traces)
The mapping $\mathsf{sTr} : \mathsf{SB} \times \mathsf{SB} \to ((\mathbf{sAct} \cup \{\checkmark\})^* \cup \mathbf{sAct}^\infty)$ *is defined by*

$$\mathsf{sTr}(\rho\|\sigma) = \begin{cases} \{\checkmark\} & \text{if } \rho = \mathbf{1} \\ \{\xi\,\boldsymbol{\chi} \mid \rho\|\sigma \overset{\xi}{\Longrightarrow} \rho'\|\sigma' \;\;\&\;\; \boldsymbol{\chi} \in \mathsf{sTr}(\rho'\|\sigma')\} & \text{if } \exists\zeta \in \mathbf{sAct}.\ \rho\|\sigma \overset{\zeta}{\Longrightarrow} \\ \{\varepsilon\} & \text{otherwise} \end{cases}$$

Let $\boldsymbol{\xi} \in \mathbf{sAct}^\infty$ *with* $\boldsymbol{\xi} = \xi_1\xi_2\ldots$. *We say* $\boldsymbol{\xi}$ *to be* definitely-skp *whenever* $\exists k.\ \forall h > k.\ \xi_h = \texttt{skp}$.

Then the notion of skp-compliance can be formalised in terms of synchronization traces as follows.

Definition 2.8 (skp-compliance) *The* client ρ *is* skip-compliant *with the* server σ, *written* $\rho \dashv^{\texttt{skp}} \sigma$, *whenever, for any* $\boldsymbol{\xi} \in \mathsf{sTr}(\rho\|\sigma)$ *either* $\boldsymbol{\xi} = \boldsymbol{\xi}'\checkmark$ *or* $\boldsymbol{\xi}$ *is infinite and not definitely*-skp.

In the remaining part of the paper we just say "compliant" instead of "skp-compliant" when any ambiguity cannot arise.

By the previous definition we have that, as stated in the Introduction, $\mathsf{Voter} \dashv^{\texttt{skp}} \mathsf{BallotServiceBehSkp}$. In the following example we provide, instead, two behaviours that are not compliant.

Example 2.9 Let us consider the following malicious server that, after receiving a login, sends a Wrong message and then, indefinitely, the message InfoW, that is

$$\mathsf{BallotServiceMalicious} \triangleq \texttt{Login}.(\overline{\texttt{Wrong}}.\mathsf{rec}\,x.\overline{\texttt{InfoW}}.x)$$

It is easy to check that $\mathsf{Voter} \not\dashv^{\mathsf{skp}} \mathsf{BallotServiceMalicious}$.
In fact we have that $\mathsf{sTr}(\mathsf{Voter} \| \mathsf{BallotServiceMalicious}) = \{\tau\,\tau\,\mathtt{skp}\,\mathtt{skp}\,\mathtt{skp}\ldots\}$, that is the only element of the set of synchronization traces is a sequence that, after the two τ actions due to the login message and the message that the login procedure went wrong, is made of an infinite number of consecutive `skp`'s, since the server would keep on skipping all the `InfoW` messages from the server. Such a sequence is an obviously *definitely*-`skp` one.

Remark 2.10 It is clear that $\dashv \subseteq \dashv^{\mathsf{skp}}$. This inclusion is strict: in fact $b \dashv^{\mathsf{skp}} \overline{a}.\overline{b}$ with $a \neq b$, but $b \not\dashv \overline{a}.\overline{b}$.

3 Coinductive characterization and decidability.

To prove that the $\dashv^{\mathsf{skp}}$ relation is decidable we work out a coinductive characterization. In doing that we use the relation synch between actions and traces. $\alpha\,\mathsf{synch}\,\sigma$ holds whenever all traces of the server σ contain the action $\overline{\alpha}$ possibly prefixed by a finite sequence of *skippable* output actions.

Definition 3.1 (Coinductive Skip-Relations)

i) The relation $\mathsf{synch} \subseteq \mathbf{Act} \times \mathsf{SB}$ *is defined by*

$$a\,\mathsf{synch}\,\sigma \triangleq \forall \boldsymbol{\alpha} \in \mathsf{Tr}(\sigma)\,\exists \overline{\boldsymbol{b}}, \boldsymbol{\alpha}'.\ \boldsymbol{\alpha} = \overline{\boldsymbol{b}}\,\overline{a}\,\boldsymbol{\alpha}' \ \ \&\ \ \overline{a} \notin \overline{\boldsymbol{b}};$$

$$\overline{a}\,\mathsf{synch}\,\sigma \triangleq \forall \boldsymbol{\alpha} \in \mathsf{Tr}(\sigma)\,\exists \overline{\boldsymbol{b}}, \boldsymbol{\alpha}'.\ \boldsymbol{\alpha} = \overline{\boldsymbol{b}}\,a\,\boldsymbol{\alpha}',$$

where $\overline{\boldsymbol{b}}$ *is possibly empty.*

ii) The operator $\mathscr{H} : \mathscr{P}(\mathsf{SB} \times \mathsf{SB}) \to \mathscr{P}(\mathsf{SB} \times \mathsf{SB})$ *is defined as follows:*
for any relation $\mathcal{R} \subseteq \mathsf{SB} \times \mathsf{SB}$, *we have* $(\rho, \sigma) \in \mathscr{H}(\mathcal{R})$ *if and only if either* $\rho \Downarrow \mathbf{1}$ *or the following statements hold:*

$$\textit{(a)}\ \ \rho \Downarrow \textstyle\sum_{i\in I} a_i.\rho_i \ \ \Rightarrow \ \ \begin{cases} \exists k \in I.\ a_k\,\mathsf{synch}\,\sigma \ \ \& \\ \forall i \in I. \forall \sigma'.\ [(a_i.\rho_i \| \sigma \stackrel{\mathtt{skp}^*\tau}{\Longrightarrow} \rho_i \| \sigma') \ \Rightarrow\ \rho_i \mathcal{R} \sigma']; \end{cases}$$

$$\textit{(b)}\ \ \rho \Downarrow \textstyle\bigoplus_{i\in I} \overline{a}_i.\rho_i \ \ \Rightarrow \ \ \begin{cases} \forall i \in I.\ a_i\,\mathsf{synch}\,\sigma \ \ \& \\ \forall i \in I. \forall \sigma'.\ [(\overline{a}_i.\rho_i \| \sigma \stackrel{\mathtt{skp}^*\tau}{\Longrightarrow} \rho_i \| \sigma') \ \Rightarrow\ \rho_i \mathcal{R} \sigma']. \end{cases}$$

iii) A relation $\mathcal{R} \subseteq \mathsf{SB} \times \mathsf{SB}$ *is a* coinductive Skip-relation *if and only if* $\mathcal{R} \subseteq \mathscr{H}(\mathcal{R})$.

Since $\mathcal{R}$ occurs covariantly in the clauses defining $\mathscr{H}(\mathcal{R})$, the operator $\mathscr{H}$ is monotonic with respect to subset inclusion. Then the following fact immediately follows by Tarsky theorem [17] (see also [16] for a discussion about the use of this result):

Fact 3.2 *Let* $\mathscr{H}^0 \triangleq \mathsf{SB} \times \mathsf{SB}$ *and* $\mathscr{H}^{k+1} \triangleq \mathscr{H}(\mathscr{H}^k)$*; then*

$$\nu(\mathscr{H}) \ = \ \bigcup\{\mathcal{R} \subseteq \mathsf{SB} \times \mathsf{SB} \mid \mathcal{R} \subseteq \mathscr{H}(\mathcal{R})\} \ = \ \bigcap_{k\in\mathbb{N}} \mathscr{H}^k$$

is the greatest fixed point of $\mathscr{H}$.

Then we define coinductively the following relation:

Definition 3.3 (Coinductive skp-Compliance)

$$\dashv^{\text{skp}}_{co.k} \triangleq \mathscr{H}^k \quad \textit{and} \quad \dashv^{\text{skp}}_{co} \triangleq \nu(\mathscr{H}),$$

where $\mathscr{H}^k$ *is defined as in Fact 3.2.*

A client ρ *is said to be* coinductively skp-compliant *with a* server σ*, whenever* $\rho \dashv^{\text{skp}}_{co} \sigma$ *holds.*

We say "coinductively compliant" as short for "coinductively skp-compliant". By the last definition we have that $\mathsf{rec}\,x.b.x \not\dashv^{\text{skp}}_{co} \mathsf{rec}\,x.\overline{a}.x$. We have as well that $b \not\dashv^{\text{skp}}_{co} \mathsf{rec}\,x.(\overline{a}.x \oplus \overline{b})$. In fact, $\overline{a}\,\overline{a}\,\overline{a}\ldots \in \mathsf{Tr}(\mathsf{rec}\,x.(\overline{a}.x \oplus \overline{b}))$ and $\neg[b \;\mathsf{synch}\; (\mathsf{rec}\,x.(\overline{a}.x \oplus \overline{b}))]$.

Proposition 3.4

$$\dashv^{\text{skp}} \;=\; \dashv^{\text{skp}}_{co}$$

Proof. [sketch] ($\subseteq$) It sufficies to show that what stated in Definition 3.1(ii) holds when we replace $\dashv^{\text{skp}}_{co}$ by $\dashv^{\text{skp}}$. In case $\rho \Downarrow \mathbf{1}$, we have that $\rho \dashv^{\text{skp}} \sigma$ immediately by definition. Let us consider the case $\rho \Downarrow \sum_{i\in I} a_i.\rho_i$. Then we observe that $\rho \dashv^{\text{skp}} \sigma$ and $\rho \not\Downarrow \mathbf{1}$ if and only if for any trace of σ there exists a prefix $\boldsymbol{a}$ such that $\sigma \overset{\boldsymbol{a}}{\Longrightarrow} \bigoplus_{h\in H} \overline{a}_h.\sigma_h$ for some $H \subseteq I$ and $\rho_h \dashv \sigma_h$ for all $h \in H$. Moreover, $\exists k \in I.\ a_k \;\mathsf{synch}\, \sigma$ holds since $\forall k \in I.\ \neg(a_k \;\mathsf{synch}\, \sigma)$ contradicts Definition 2.8. If $\rho \Downarrow \bigoplus_{i\in I} \overline{a}_i.\rho_i$ the proof proceeds in a similar way.

($\supseteq$) Let us assume $\rho \not\dashv^{\text{skp}} \sigma$. This implies that $\neg(\rho \Downarrow \mathbf{1})$. Then, by Definition 2.8, there exists $\boldsymbol{\xi} \in \mathsf{sTr}(\rho\|\sigma)$ such that either $\boldsymbol{\xi}$ is finite but $\boldsymbol{\xi} \neq \boldsymbol{\xi}'\checkmark$ for any $\boldsymbol{\xi}'$, or $\boldsymbol{\xi}$ is infinite and definitely-skp. In the first case we proceed by induction on the lenght of the τ-actions in $\boldsymbol{\xi}$ to contradict condition $\exists k \in I.\ a_k \;\mathsf{synch}\, \sigma$ in Definition 3.1(ii). In the infinite case, we get a contradiction to the
$\forall i \in I. \forall \sigma'.\ [(\alpha_i.\rho_i \| \sigma \overset{\text{skp}^*\tau}{\Longrightarrow} \rho_i \| \sigma') \;\Rightarrow\; \rho_i \mathscr{R} \sigma']$ (for the proper α_i) clauses in Definition 3.1(ii). $\square$

It is possible to show the relation $\dashv^{\text{skp}}$ to be decidable. In order to do that we define a formal system that reflects the coinductive definition of the $\dashv^{\text{skp}}$ relation, and whose derivation rules can be looked at as rules of a recursive, syntax-driven decision algorithm, where the decision process coincides with a proof reconstruction procedure.

In the formal system, the assumptions in an environment are actually *marked* assumptions. The markings are used to prevent the possibility of getting a correct derivation for compliance statements that allow for definitely-skp client$\|$server interactions.

Definition 3.5 (A formal system for $\dashv^{\text{skp}}$)

i) A marked environment Γ *is a finite set of marked assumptions of the form* $(\rho' \dashv^{\text{skp}} \sigma')_\bullet$*, where* $\rho', \sigma' \in \mathsf{SB}$ *and* $\bullet \in \{\mathtt{ok}, \mathtt{no}\}$.

ii) A judgment is an expression of the form $\Gamma \rhd \rho \dashv^{\text{skp}} \sigma$*, where* Γ *is a marked environment. The axioms and rules of the system deriving judgments are in Figure 1, where the environment* $\Gamma_{\mathtt{ok}}$ *is defined by* $\Gamma_{\mathtt{ok}} = \{(\rho' \dashv^{\text{skp}} \sigma')_{\mathtt{ok}} \mid (\rho' \dashv^{\text{skp}} \sigma')_{\mathtt{ok}} \in \Gamma \vee (\rho' \dashv^{\text{skp}} \sigma')_{\mathtt{no}} \in \Gamma\}$.

We assume any marked environment to be *coherent*, that is there can be no two assumptions with the same compliance statement and different markings in the same environment, like $(\rho \dashv^{\text{skp}} \sigma)_{\mathtt{ok}}$ and $(\rho \dashv^{\text{skp}} \sigma)_{\mathtt{no}}$. Moreover, it will be easy to check that the derivation reconstruction procedure always produce coherent environments.

The intended meaning of a judgment $\Gamma \rhd \rho \dashv^{\text{skp}} \sigma$ is that if, for any $(\rho' \dashv^{\text{skp}} \sigma')_\bullet \in \Gamma$, $\rho' \dashv^{\text{skp}} \sigma'$ holds, then $\rho \dashv^{\text{skp}} \sigma$ holds as well, except for some judgments for which the interaction between ρ and

$$\frac{}{\Gamma \rhd \mathbf{1} \dashv^{\mathsf{skp}} \sigma}\ (\textsc{Ax}) \qquad \frac{}{\Gamma, (\rho \dashv^{\mathsf{skp}} \sigma)_{\mathtt{ok}} \rhd \rho \dashv^{\mathsf{skp}} \sigma}\ (\textsc{Hyp})$$

$$\frac{\Gamma \rhd \sigma\{\mathsf{rec}\, x.\sigma/x\} \dashv^{\mathsf{skp}} \sigma'}{\Gamma \rhd \mathsf{rec}\,.\sigma \dashv^{\mathsf{skp}} \sigma'}\ (\textsc{Unf-L}) \qquad \frac{\Gamma \rhd \sigma' \dashv^{\mathsf{skp}} \sigma\{\mathsf{rec}\, x.\sigma/x\}}{\Gamma \rhd \sigma' \dashv^{\mathsf{skp}} \mathsf{rec}\, x.\sigma}\ (\textsc{Unf-R})$$

$$\frac{\forall i \in (I \setminus K).\ \Gamma' \rhd \sum_{k\in K} a_k.\rho_k \dashv^{\mathsf{skp}} \sigma_i \qquad \forall j \in (K \cap I).\ \Gamma'' \rhd \rho_j \dashv^{\mathsf{skp}} \sigma_j}{\Gamma \rhd \sum_{k\in K} a_k.\rho_k \dashv^{\mathsf{skp}} \bigoplus_{i\in I} \overline{a}_i.\sigma_i}\ (+.\oplus\textsc{-Cpl})$$

$$\text{where } \Gamma' = \Gamma,\ (\textstyle\sum_{k\in K} a_k.\rho_k \dashv^{\mathsf{skp}} \bigoplus_{i\in I} \overline{a}_i.\sigma_i)_{\mathtt{no}}$$
$$\Gamma'' = \Gamma_{\mathtt{ok}},\ (\textstyle\sum_{k\in K} a_k.\rho_k \dashv^{\mathsf{skp}} \bigoplus_{i\in I} \overline{a}_i.\sigma_i)_{\mathtt{ok}}$$

$$\frac{\forall i \in I.\ \Gamma' \rhd \bigoplus_{k\in K} \overline{a}_k.\rho_k \dashv^{\mathsf{skp}} \sigma_i}{\Gamma \rhd \bigoplus_{k\in K} \overline{a}_k.\rho_k \dashv^{\mathsf{skp}} \bigoplus_{i\in I} \overline{b}_i.\sigma_i}\ (\oplus.\oplus\textsc{-Cpl}) \qquad \frac{K \subseteq I \qquad \forall k \in K.\ \Gamma' \rhd \rho_k \dashv^{\mathsf{skp}} \sigma_k}{\Gamma \rhd \bigoplus_{k\in K} \overline{a}_k.\rho_k \dashv^{\mathsf{skp}} \sum_{i\in I} a_i.\sigma_i}\ (\oplus.+\textsc{-Cpl})$$

$$\text{where } \Gamma' = \Gamma,\ (\textstyle\bigoplus_{k\in K} \overline{a}_k.\rho_k \dashv^{\mathsf{skp}} \bigoplus_{i\in I} \overline{b}_i.\sigma_i)_{\mathtt{no}} \qquad \text{where } \Gamma' = \Gamma_{\mathtt{ok}},\ (\textstyle\bigoplus_{k\in K} \overline{a}_k.\rho_k \dashv^{\mathsf{skp}} \sum_{i\in I} a_i.\sigma_i)_{\mathtt{ok}}$$

Figure 1: The formal system $\rhd$ for $\dashv^{\mathsf{skp}}$

σ would produce definitely-`skp` syncrhonization traces. The use of markings rules out such a possiblity. In fact the following Soundness and Completeness result we obtain is, as needed, for derivations with empty environment.

Theorem 3.6 (Soundness and Completeness)

$$\rho \dashv^{\mathsf{skp}} \sigma \quad \Leftrightarrow \quad \emptyset \rhd \rho \dashv^{\mathsf{skp}} \sigma.$$

The proof of the Soundness and Completeness property above can be obtained by first proving it for a system without markings and allowing for definitely-`skp` interaction sequences in the definition of compliance. And then by showing that such definitely-`skp` sequences are ruled out if the derivations are properly marked. The proof of the first part can be obtained along the lines used in [4] for a similar system.

Theorem 3.7 (Decidability of $\rhd$) *The system of Figure 1 is decidable.*

Proof. [Sketch] The system in Figure 1 satisfies a sort of subformula property. As a matter of fact the behaviours used in the premises of any rule are subterms (for a suitable and natural definition of subterm) of those used in the premises of the rule. This implies the system to be algorithmic: a decision procedure consists in a breadth first searching for a proof of a judgment in a bottom-up, syntax-driven, way. Such proof reconstruction ends since, for any possible branch of the proof, we eventually find either (1) an axiom $(\textsc{Ax})$ or (2) an hypothesis $(\textsc{Hyp})$ or (3) a wrong hypothesis, that is a judgment of the form $\Gamma,(\rho \dashv^{\mathsf{skp}} \sigma)_{\mathtt{no}} \rhd \rho \dashv^{\mathsf{skp}} \sigma$ or, by the subformula property, (4) a previously encountered judgment. In case (3) or (4) are encountered along a branch, the derivation reconstruction algorithm fails. In particular, the presence of (3) denotes the possibility of a definitely-`skp` synchronization trace. Notice that the proof reconstruction is also deterministic, but possibly for the choice of the order in which (Unf-L) and (Unf-R) occur along a branch in the proof tree, which is immaterial as they have to be consecutive. The complete proof develops along the same lines used for a similar proof in [4], where we resort to a similar argument used in [15] and thereafter in [10]. □

Decidability of compliance is now easily got as a corollary.

Corollary 3.8 *The relation $\dashv^{\mathsf{skp}}$ is decidable.*

Proof. By Theorems 3.6 and 3.7. □

In the following we provide two simple example of application of the syntax-driven derivation reconstruction algorithm described in the proof of Theorem 3.7. In the first example the algorithm fails because one of the possible interaction sequences for the client b and the server $\mathsf{rec}\,x.(\overline{a}.x \oplus \overline{b})$ interaction would be definitely-`skp`. In the second one the algorithm succeeds and produce the right derivation. Notice how the failure in the first example is due to the fact that along the leftmost branch we encounter a *wrong hypothesis*, that is a judgment of the form $\Gamma,(\rho \dashv^{\mathsf{skp}} \sigma)_{\mathtt{no}} \rhd \rho \dashv^{\mathsf{skp}} \sigma$.

Example 3.9 *Given $\sigma = \mathsf{rec}\,x.(\overline{a}.x \oplus \overline{b})$, the reconstruction algorithm for $\rhd\, b \dashv^{\mathsf{skp}} \sigma$ produces the following result:*

$$\dfrac{\dfrac{\dfrac{\dfrac{\begin{array}{c}\textbf{FAIL!}\\ (b \dashv^{\mathsf{skp}} \overline{a}.\sigma \oplus \overline{b})_{\mathtt{no}} \rhd b \dashv^{\mathsf{skp}} \overline{a}.\sigma \oplus \overline{b}\end{array}}{(b \dashv^{\mathsf{skp}} \overline{a}.\sigma \oplus \overline{b})_{\mathtt{no}} \;\rhd\; b \dashv^{\mathsf{skp}} \mathsf{rec}\,x.(\overline{a}.x \oplus \overline{b})}\,(\textsc{Unf-R}) \qquad \dfrac{}{(b \dashv^{\mathsf{skp}} \overline{a}.\sigma \oplus \overline{b})_{\mathtt{ok}} \;\rhd\; \mathbf{1} \dashv^{\mathsf{skp}} \mathbf{1}}\,(\textsc{Ax})}{\rhd\; b \dashv^{\mathsf{skp}} \overline{a}.\sigma \oplus \overline{b}}\,(+.\oplus\text{-}\textsc{Cpl})}{\rhd\; b \dashv^{\mathsf{skp}} \mathsf{rec}\,x.(\overline{a}.x \oplus \overline{b})}\,(\textsc{Unf-R})}{}$$

Example 3.10 *Given $\rho = \mathsf{rec}\,x.b.x$ and $\sigma = \mathsf{rec}\,y.(\overline{a}.\overline{b}.\overline{a}.y \oplus \overline{b}.\mathsf{rec}\,x.\overline{b}.x)$, the reconstruction algorithm for $\rhd\, \rho \dashv^{\mathsf{skp}} \sigma$ produces the following result:*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(\gamma_1)_{\mathtt{ok}},(\gamma_2)_{\mathtt{ok}},(\gamma_3)_{\mathtt{ok}},(\gamma_4)_{\mathtt{no}} \;\rhd\; \gamma_2}\,(\textsc{Hyp})}{(\gamma_1)_{\mathtt{ok}},(\gamma_2)_{\mathtt{ok}},(\gamma_3)_{\mathtt{ok}},(\gamma_4)_{\mathtt{no}} \;\rhd\; b.\rho \dashv^{\mathsf{skp}} \sigma}\,(\textsc{Unf-R})}{(\gamma_1)_{\mathtt{ok}},(\gamma_2)_{\mathtt{ok}},(\gamma_3)_{\mathtt{ok}} \;\rhd\; b.\rho \dashv^{\mathsf{skp}} \overline{a}.\sigma}\,(+.\oplus\text{-}\textsc{Cpl})}{(\gamma_1)_{\mathtt{ok}},(\gamma_2)_{\mathtt{ok}},(\gamma_3)_{\mathtt{ok}} \;\rhd\; \rho \dashv^{\mathsf{skp}} \overline{a}.\sigma}\,(\textsc{Unf-L})}{(\gamma_1)_{\mathtt{no}},(\gamma_2)_{\mathtt{no}} \;\rhd\; b.\rho \dashv^{\mathsf{skp}} \overline{b}.\overline{a}.\sigma}\,(+.\oplus\text{-}\textsc{Cpl}) \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{(\gamma_1)_{\mathtt{ok}},(\gamma_2)_{\mathtt{ok}},(\gamma_5)_{\mathtt{ok}} \;\rhd\; \gamma_5}\,(\textsc{Hyp})}{(\gamma_1)_{\mathtt{ok}},(\gamma_2)_{\mathtt{ok}},(\gamma_5)_{\mathtt{ok}} \;\rhd\; \rho \dashv^{\mathsf{skp}} \rho}\,(\textsc{Unf-L-R})}{(\gamma_1)_{\mathtt{no}},(\gamma_2)_{\mathtt{no}} \;\rhd\; b.\rho \dashv^{\mathsf{skp}} \overline{b}.\rho}\,(+.\oplus\text{-}\textsc{Cpl})}{(\gamma_1)_{\mathtt{no}},(\gamma_2)_{\mathtt{no}} \;\rhd\; \rho \dashv^{\mathsf{skp}} \mathsf{rec}\,x.\overline{b}.x}\,(\textsc{Unf-L-R})}{(\gamma_1)_{\mathtt{no}} \;\rhd\; b.\rho \dashv^{\mathsf{skp}} \overline{a}.\overline{b}.\overline{a}.\sigma \oplus \overline{b}.\mathsf{rec}\,x.\overline{b}.x)}\,(+.\oplus\text{-}\textsc{Cpl})}{(\gamma_1)_{\mathtt{no}} \;\rhd\; b.\rho \dashv^{\mathsf{skp}} \sigma}\,(\textsc{Unf-R})}{\rhd\; b.\rho \dashv^{\mathsf{skp}} \overline{c}.\sigma}\,(+.\oplus\text{-}\textsc{Cpl})}{\rhd\; \mathsf{rec}\,x.b.x \dashv^{\mathsf{skp}} \overline{c}.\mathsf{rec}\,y.(\overline{a}.\overline{b}.\overline{a}.y \oplus \overline{b}.\mathsf{rec}\,x.\overline{b}.x)}\,(\textsc{Unf-L})$$

where
$\gamma_1 = b.\rho \dashv^{\mathsf{skp}} \overline{c}.\sigma$
$\gamma_2 = b.\rho \dashv^{\mathsf{skp}} \overline{a}.\overline{b}.\overline{a}.\sigma \oplus \overline{b}.\mathsf{rec}\,x.\overline{b}.x$
$\gamma_3 = b.\rho \dashv^{\mathsf{skp}} \overline{b}.\overline{a}.\sigma$
$\gamma_4 = b.\rho \dashv^{\mathsf{skp}} \overline{a}.\sigma$
$\gamma_5 = b.\rho \dashv^{\mathsf{skp}} \overline{b}.\rho$

4 The skp-subbehaviour relation

As mentioned in the Introduction, in the theory of contracts the compliance relation induces a preorder $\preceq$. The relation $\sigma \preceq \sigma'$ holds whenever, for any client ρ, if $\rho \dashv \sigma$ then $\rho \dashv \sigma'$.

If σ, σ' and ρ are required to be in SB then this relation, which we call *subbehaviour relation* (dubbed $\preceq_s$ in [2]), coincides with the testing must-preorder [13], which is not the case if arbitrary contracts are considered (see [5]). Here we relativize the definition of the subbehavior relation to the $\dashv^{\mathtt{skp}}$ relation studied in the previous section, obtaining a new relation, which we call skp-*subbehaviour* and dub $\preceq^{\mathtt{skp}}$.

Definition 4.1 (skp-Subbehaviour)
Over SB *it is defined the binary relation* $\sigma \preceq^{\mathtt{skp}} \sigma'$ *by*

$$\sigma \preceq^{\mathtt{skp}} \sigma' \;\;\Leftrightarrow\;\; \forall \rho \in \mathsf{SB}.\ [\rho \dashv^{\mathtt{skp}} \sigma \Rightarrow \rho \dashv^{\mathtt{skp}} \sigma'].$$

Remark 4.2 It is not difficult to check that $\preceq^{\mathtt{skp}} \not\subseteq \preceq$ by means of the following easy counterexample. We have that $a \preceq^{\mathtt{skp}} \overline{c}.a$. In fact all the possible skp-compliant clients of a are $\{\mathbf{1}, \overline{a}\}$, which are trivially also skp-compliant with the server $\overline{c}.a$ by skipping the action $\overline{c}$. Without the possibilty of skipping such an action, we have that $\overline{a}$ is not a client of $\overline{c}.a$ anymore, whereas it is still so of a. That is $a \not\preceq \overline{c}.a$.
For what concern the opposite inclusion, we conjecture it to hold. Of course the proof would not be immediate. By Remark 2.10, we know that $\dashv \subset \dashv^{\mathtt{skp}}$, but this fact doesn't imply that $\preceq$ is included in $\preceq^{\mathtt{skp}}$, because $\sigma \preceq^{\mathtt{skp}} \sigma'$ depends on a negative occurrence of the hypothesis $\rho \dashv^{\mathtt{skp}} \sigma$.

4.1 *Duals as minima* property and decidability of $\preceq^{\mathtt{skp}}$

We proceed now towards the proof of decidability of the skp-subbehaviour relation. This will be obtained as a corollary of the property that the dual of a session-behaviour is actually the minimum among its servers w.r.t. $\preceq^{\mathtt{skp}}$. For any theory of subcontracts this *duals as minima* result is quite relevant, since the possibility of implementing contract-based query engines relies on it. This is well explained in [14] in the paragraph that we quote below.

> Formal notions of compliance and subcontract relation may be used for implementing contract-based query engines. The query for services that satisfy ρ is answered with the set $\mathcal{Q}_1(\rho) = \{\sigma \mid \rho \dashv \sigma\}$. The complexity of running this query grows with the number of services stored in the repository. A better strategy is to compute the dual contract of ρ, denoted by $\rho^{\perp}$ [$\overline{\rho}$ *in our context*], which represents the canonical service satisfying ρ (that is $\rho \dashv \rho^{\perp}$) and then answering the query with the set $\mathcal{Q}_2(\rho) = \{\sigma \mid \sigma \preceq \rho^{\perp}\}$. If $\rho^{\perp}$ is the $\preceq$-smallest service that satisfies ρ, we have $\mathcal{Q}_1(\rho) = \mathcal{Q}_2(\rho)$, namely we are guaranteed that no service is mistakenly excluded. The advantage of this approach is that $\preceq$ can be precomputed when services are registered in the repository, and the query engine needs only scan through the $\preceq$-minimal contracts.
> (*L.Padovani* - [14], Sect.1)

The minimum property of dual behaviours can be proved using the following property:

$$\rho \dashv^{\mathtt{skp}} \gamma \ \&\ \overline{\gamma} \dashv^{\mathtt{skp}} \sigma \;\;\Rightarrow\;\; \rho \dashv^{\mathtt{skp}} \sigma \qquad (1)$$

This property, however, is not easy to establish in presence of skipped actions, as exemplified in the following. It is immediate to check that:

$$a.d \dashv^{\mathtt{skp}} \overline{b}.\overline{b}.\overline{a}.\overline{d} \qquad \& \qquad b.b.a.d \dashv^{\mathtt{skp}} \overline{a}.\overline{b}.\overline{a}.\overline{b}.\overline{a}.\overline{d}$$

Each b in $b.b.a.d$ skips an $\overline{a}$ before synchronizing with its dual $\overline{b}$, whereas the action that synchronizes with the a in $b.b.a.d$ is the last $\overline{a}$ of $\overline{a}.\overline{b}.\overline{a}.\overline{b}.\overline{a}.\overline{d}$. Now, the a in $a.d$ synchronizes after skipping the two $\overline{b}$'s corresponding to the first two b's of $b.b.a.d$. The action in $\overline{a}.\overline{b}.\overline{a}.\overline{b}.\overline{a}.\overline{d}$ synchronizing with the a in $a.d$, however, it is not the last $\overline{a}$ of $\overline{a}.\overline{b}.\overline{a}.\overline{b}.\overline{a}.\overline{d}$, but actually the first one.

This fact, fortunately, does not cause any problem for $a.d \dashv^{\mathtt{skp}} \overline{a}.\overline{b}.\overline{a}.\overline{b}.\overline{a}.\overline{d}$ since the d in $a.d$ synchronizes with the $\overline{d}$ of $\overline{a}.\overline{b}.\overline{a}.\overline{b}.\overline{a}.\overline{d}$ by skipping all the actions $\overline{b}.\overline{a}.\overline{b}.\overline{a}$ between the first $\overline{a}$ and $\overline{d}$. The presence of cases like these require to be carefully handled when proving property (1) that is otherwise similar to the analogous facts in [2, 4].

To ease the proof we first consider an equivalent formulation of the skp-compliance relation. We introduce a relation $\sqsubseteq$ between sequences of actions, such that $a_1 \dots a_n \sqsubseteq b_1 \dots b_m$ holds whenever any a_i (going from left to right) coincides with some b_j, provided that all the elements between the element b_h coinciding with a_{i-1} and b_j are distinct from b_j. For instance, $bbad \sqsubseteq abababd$ and $ad \sqsubseteq abababd$, whereas $ad \not\sqsubseteq bbaa$.

Definition 4.3 (The $\sqsubseteq$ relation.)

i) *The binary relation $\sqsubseteq \subseteq \mathcal{N}^+ \times \mathcal{N}^+$ on finite and non empty sequences of input actions is inductively defined as follows.*
Let $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{N}^+$.

- $b \sqsubseteq a_1 \dots a_k b \;\triangleq\; k \geq 0 \;\&\; b \neq a_1, \dots, a_k$
- $b\,\boldsymbol{a} \sqsubseteq a_1 \dots a_k b \boldsymbol{b} \;\triangleq\; \boldsymbol{a} \sqsubseteq \boldsymbol{b} \;\&\; k \geq 0 \;\&\; b \neq a_1, \dots, a_k$

ii) *The above relation is naturally extended to $\mathcal{N}^\infty \times \mathcal{N}^\infty$ and to $\mathcal{N}^+ \times \mathcal{N}^\infty$*

The relation $\sqsubseteq$ will be used in the alternative coinductive skp-compliance provided in Lemma 4.5 below. It will be used to represent, on the left-hand side, the synchronizing actions of the client and on the right-hand side the corresponding actions of the server, possibly preceded by a finite number of *skipped* actions. The relation is extended to $\mathcal{N}^\infty \times \mathcal{N}^\infty$ since a client can be skp-compliant with a server even without ever terminating. It is extended to $\mathcal{N}^+ \times \mathcal{N}^\infty$ since a client can succesfully terminate even if its server could be able to going on indefinitely.

The following property holds for $\sqsubseteq$.

Lemma 4.4 *The relation $\sqsubseteq$ is transitive.*

Lemma 4.5 (Alternative coinductive skp-compliance)

$$\nu(\mathcal{H}) = \nu(\mathcal{J})$$

where the operator $\mathcal{J} : \mathcal{P}(\mathsf{SB} \times \mathsf{SB}) \to \mathcal{P}(\mathsf{SB} \times \mathsf{SB})$ is defined as follows: for any relation $\mathcal{R} \subseteq \mathsf{SB} \times \mathsf{SB}$, $(\rho, \sigma) \in \mathcal{J}(\mathcal{R})$ if and only if either $\rho \Downarrow \mathbf{1}$ or, whenever $[\rho \Downarrow\!\!\!\!/ \sum_{i \in I} a_i.\rho_i \;\&\; \sigma \Downarrow\!\!\!\!/ \sum_{j \in J} a_j.\sigma_j]$, the following statements hold:

$$a)\;\; \rho \Downarrow \textstyle\sum_{i \in I} a_i.\rho_i \;\;\Rightarrow\;\; \begin{cases} \{\overline{\boldsymbol{b}} \mid |\boldsymbol{b}| > 0,\; \sigma \stackrel{\overline{\boldsymbol{b}}}{\Longrightarrow}\} \neq \emptyset \\[2ex] \forall \overline{\boldsymbol{b}} \text{ s.t. } \sigma \stackrel{\overline{\boldsymbol{b}}}{\Longrightarrow} \sigma'.\;\; \exists \boldsymbol{a} \sqsubseteq \boldsymbol{b}.\;\; (\rho \underset{max}{\stackrel{\boldsymbol{a}}{\Longrightarrow}} \rho' \;\&\; \rho' \,\mathcal{J}\, \sigma') \\[2ex] \forall \overline{\boldsymbol{b}}^\infty \in \mathsf{Tr}(\sigma).\; \exists \boldsymbol{a} \sqsubseteq \boldsymbol{b}.\;\; (\rho \underset{max}{\stackrel{\boldsymbol{a}}{\Longrightarrow}} \mathbf{1} \;\vee\; \rho \underset{max}{\stackrel{\boldsymbol{a}^\infty}{\Longrightarrow}}) \end{cases}$$

$$b)\;\; \rho \Downarrow \textstyle\bigoplus_{i \in I} \overline{a}_i.\rho_i \;\;\Rightarrow\;\; \begin{cases} \{\overline{\boldsymbol{a}}^\infty \mid \sigma \stackrel{\overline{\boldsymbol{a}}^\infty}{\Longrightarrow}\} = \emptyset \\[2ex] \forall j \in I\,.\forall \boldsymbol{a} \text{ s.t. } \sigma \underset{max}{\stackrel{\overline{\boldsymbol{a}}}{\Longrightarrow}}\;.\; (\sigma \stackrel{\overline{\boldsymbol{a}} a_j}{\Longrightarrow} \sigma' \;\&\; \rho_j \,\mathcal{J}\, \sigma') \end{cases}$$

where $\sigma \overset{\boldsymbol{a}}{\underset{max}{\Longrightarrow}} \triangleq [\sigma \overset{\boldsymbol{a}}{\Longrightarrow} \& \nexists c \in \mathcal{N}.\, \sigma \overset{\boldsymbol{a}c}{\Longrightarrow}]$ *and* $\sigma \overset{\overline{\boldsymbol{a}}}{\underset{max}{\Longrightarrow}} \triangleq [\sigma \overset{\overline{\boldsymbol{a}}}{\Longrightarrow} \& \nexists \overline{c} \in \overline{\mathcal{N}}.\, \sigma \overset{\overline{\boldsymbol{a}}\overline{c}}{\Longrightarrow}]$.

The following property will be useful to show the dual-as-minimum property.

Lemma 4.6 *Given* $\sigma \overset{\overline{\boldsymbol{a}}}{\underset{max}{\Longrightarrow}}$ *with* $\gamma \dashv^{\mathrm{skp}} \sigma$, *there exists* $\boldsymbol{c}$ *s.t.* $\gamma \overset{\boldsymbol{c}}{\underset{max}{\Longrightarrow}} \gamma'$ & $\gamma' \Downarrow \bigoplus \overline{b}_j.\gamma'_j$. *Moreover, for any* $\overline{b}_j$ *we have* $\sigma \overset{\overline{\boldsymbol{a}}\, b_j}{\Longrightarrow} \sigma'_j$ *with* $\gamma'_j \dashv^{\mathrm{skp}} \sigma'_j$.

Lemma 4.7 *For all* $\rho, \sigma, \gamma \in \mathsf{SB}$*: if* $\rho \dashv^{\mathrm{skp}} \gamma$ *and* $\overline{\gamma} \dashv^{\mathrm{skp}} \sigma$ *then* $\rho \dashv^{\mathrm{skp}} \sigma$.

Proof. By Lemma 3.4, we have to prove the relation $\mathcal{K} = \{(\rho,\sigma) \mid \exists\gamma.\ \rho \dashv^{\mathrm{skp}} \gamma\ \&\ \overline{\gamma} \dashv^{\mathrm{skp}} \sigma\}$ to be a coinductive Skip-compliance. We shall do that by using the alternative characterization of coinductive `skp`-compliance of Lemma 4.5. Let ρ and σ be such that $\rho \dashv^{\mathrm{skp}} \gamma$ and $\overline{\gamma} \dashv^{\mathrm{skp}} \sigma$ for some γ. There are two cases, of which we consider the most complex one for lack of space.

$\rho \Downarrow \bigoplus_{i\in I} \overline{a}_i.\rho_i$: Let $k \in I$. From $\rho \dashv^{\mathrm{skp}} \gamma$, 4.5, we get that $\{\overline{\boldsymbol{c}}^{\infty} \mid \gamma \overset{\overline{\boldsymbol{c}}^{\infty}}{\Longrightarrow}\} = \emptyset$ and that,

$$\forall\, \overline{c}\ s.t.\ \gamma \overset{\overline{\boldsymbol{c}}}{\underset{max}{\Longrightarrow}} \gamma''\ .\ \ \gamma'' \Downarrow \sum b_j.\gamma'_j\ \ \&\ \ \exists h.\ b_h \equiv a_k\ \ \&\ \ \rho_k \dashv^{\mathrm{skp}} \gamma'_h. \tag{2}$$

By duality, we get that for all $\boldsymbol{c}$ s.t. $\overline{\gamma} \overset{\boldsymbol{c}}{\underset{max}{\Longrightarrow}} \overline{\gamma''}$, $\overline{\gamma''} \Downarrow \bigoplus \overline{b}_j.\overline{\gamma'}_j$. We can now infer that $\{\overline{\boldsymbol{a}}^{\infty} \mid \sigma \overset{\overline{\boldsymbol{a}}^{\infty}}{\Longrightarrow}\} = \emptyset$, by distinguishing two cases: if $\gamma \Downarrow \bigoplus \overline{c}_p.\gamma_p$, it is immediate by $\overline{\gamma} \dashv^{\mathrm{skp}} \sigma$ and Lemma 4.5. Otherwise, by contradiction, let assume that there exists $\boldsymbol{a}^{\infty}$ such that $\sigma \overset{\overline{\boldsymbol{a}}^{\infty}}{\Longrightarrow}$. By $\overline{\gamma} \dashv^{\mathrm{skp}} \sigma$ and Lemma 4.5 we get that there exists $\boldsymbol{d} \sqsubseteq \boldsymbol{a}$ such that either $\overline{\gamma} \overset{\boldsymbol{d}}{\underset{max}{\Longrightarrow}}$ or $\overline{\gamma} \overset{\boldsymbol{d}^{\infty}}{\underset{max}{\Longrightarrow}}$. We obtain an immediate contradiction in the second case, whereas in the first one, we get a contradiction by the fact that $\gamma \overset{\overline{\boldsymbol{d}}}{\underset{max}{\Longrightarrow}} \mathbf{1}$ and by (2). Now, given $\sigma \overset{\overline{\boldsymbol{a}}}{\underset{max}{\Longrightarrow}}$, from $\overline{\gamma} \dashv^{\mathrm{skp}} \sigma$ and Lemma 4.6, given $\sigma \overset{\overline{\boldsymbol{a}}}{\underset{max}{\Longrightarrow}}$, there exists $\boldsymbol{c}'$ s.t. $\overline{\gamma} \overset{\boldsymbol{c}'}{\underset{max}{\Longrightarrow}} \overline{\gamma''_1}$, $\overline{\gamma''_1} \Downarrow \bigoplus \overline{b}_j.\overline{\gamma'_1}_j$, moreover, for any $\overline{b}_j$ we have $\sigma \overset{\overline{\boldsymbol{a}}\, b_j}{\Longrightarrow} \sigma'_j$ with $\overline{\gamma'_1}_j \dashv^{\mathrm{skp}} \sigma'_j$. From (2) we get that

$$\gamma \overset{\overline{\boldsymbol{c}'}}{\underset{max}{\Longrightarrow}} \gamma''\ .\ \ \gamma''_1 \Downarrow \sum b_j.{\gamma_1}'_j\ \ \&\ \ \exists h.\ b_h \equiv a_k\ \ \&\ \ \rho_k \dashv^{\mathrm{skp}} {\gamma_1}'_h.$$

Since $\overline{\gamma'_1}_h \dashv^{\mathrm{skp}} \sigma'_h$, we get $\rho_k \mathcal{K} \sigma'_h$.

□

Proposition 4.8 (Duals as minima)
Let $\rho \in \mathsf{SB}$. *Then* $\overline{\rho}$ *is the minimum server of* ρ, *i.e. :* $\quad \forall \sigma.\ \rho \dashv^{\mathsf{skp}} \sigma \ \Rightarrow\ \overline{\rho} \preceq^{\mathsf{skp}} \sigma$

Proof. Let σ and γ be such that $\rho \dashv^{\mathsf{skp}} \sigma$ and $\gamma \dashv^{\mathsf{skp}} \overline{\rho}$. It is immediate to check that $\overline{\rho} \dashv^{\mathsf{skp}} \rho$. Hence, by Lemma 4.7 and the fact that the $\bar{\cdot}$ operation is involutive, we have that $\gamma \dashv^{\mathsf{skp}} \sigma$, so showing that $\overline{\rho} \preceq^{\mathsf{skp}} \sigma$. □

We are finally in place to establish the following result.

Theorem 4.9 $\quad \sigma \preceq^{\mathsf{skp}} \sigma' \quad \Longleftrightarrow \quad \overline{\sigma} \dashv^{\mathsf{skp}} \sigma'$

Proof. ($\Rightarrow$) Let $\overline{\sigma} \not\dashv^{\mathsf{skp}} \sigma'$. Since we have $\overline{\sigma} \dashv^{\mathsf{skp}} \sigma$, we get then that $\sigma \not\preceq^{\mathsf{skp}} \sigma'$.
($\Leftarrow$) Let $\overline{\sigma} \dashv^{\mathsf{skp}} \sigma'$. Then, by Proposition 4.8, we get $\sigma = \overline{\overline{\sigma}} \preceq^{\mathsf{skp}} \sigma'$. □

By Theorem 4.9 and decidability of $\dashv^{\mathsf{skp}}$ stated in Corollary 3.8 we conclude:

Corollary 4.10 *The relation* $\preceq^{\mathsf{skp}}$ *is decidable.*

5 Related works

What we devised in the present paper is not the only possibility of weakening the notions of compliance and sub-behaviour. An alternative approach in the setting of (first-order and unrestricted) contracts has been followed by Luca Padovani in [14].

We briefly recall Padovani's approach to compare with ours, which is possible because session-behaviours are particular contracts. In [14] the interactions between a client and a server can be mediated (coordinated) by an *orchestrator*, a particular process (a sort of *active channel* or *channel controller*) with the capability of buffering messages. Thanks to that, the server's "answers" to the client's "requests" can be delivered in a different order, so enabling a form of asynchronous interactions, or kept indefinitely in the buffer, that is equivalent to *discarding* them. The weak-compliance relation resulting from the presence of orchestrators, that we denote here by $\dashv^{\mathsf{P}}$, induces a preorder that is also investigated in [14], and that here we refer to as $\preceq^{\mathsf{P}}$.

Let us explain the use of orchestrators by means of an example. The following is the behaviour of a ballot service similar to one we already described in the Introduction. Logging-in can be retried in case of wrong login or system overload. The message `Id` denotes the identifier of the transaction provided by the server to its clients.

$$\begin{aligned}&\mathsf{BallotServiceBehP} \triangleq\\ &\qquad \mathsf{rec}x.\,\mathtt{Login}.(\overline{\mathtt{Wrong}}.x \oplus \overline{\mathtt{Overload}}.x \oplus \overline{\mathtt{Ok}}.\overline{\mathtt{Id}}.(\ \ \mathtt{VoteA}.(\mathtt{Va1}+\mathtt{Va2})\\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad + \mathtt{VoteB}.(\mathtt{Vb1}+\mathtt{Vb2})\ )\ )\end{aligned}$$

Now, let us assume to have a voter with the following behaviour:

$$\mathsf{VoterBehP} \triangleq \mathsf{rec}x.\,\overline{\mathtt{Login}}.(\mathtt{Wrong}.x + \mathtt{Overload}.x + \mathtt{Ok}.\overline{\mathtt{Vb1}}.\overline{\mathtt{VoteB}})$$

Such a voter, besides not needing any identifier of the transaction, intends to give the preference for the vice-candidate *before* the one for the main candidate. The feasibility of the interaction between $\mathsf{VoterBehP}$ and $\mathsf{BallotServiceBehP}$ can be guaranteed only by the presence of an orchestrator such as:

$$\begin{aligned}\mathsf{BallotOrchP} \triangleq \mathsf{rec}x.\,\langle\mathtt{Login},\overline{\mathtt{Login}}\rangle.(\ \ &\langle\overline{\mathtt{Wrong}},\mathtt{Wrong}\rangle.x\\ \vee\ &\langle\overline{\mathtt{Overload}},\mathtt{Overload}\rangle.x\\ \vee\ &\langle\overline{\mathtt{Ok}},\mathtt{Ok}\rangle.\langle\varepsilon,\mathtt{Id}\rangle.\langle\mathtt{Vb1},\varepsilon\rangle.\langle\mathtt{VoteB},\overline{\mathtt{VoteB}}\rangle.\langle\varepsilon,\overline{\mathtt{Vb1}}\rangle)\end{aligned}$$

The actions of an orchestrator are actually pairs. The first orchestrating action $\langle \texttt{Login}, \overline{\texttt{Login}} \rangle$ means that BallotOrch immediately delivers to the server a login, represented by the action $\overline{\texttt{Login}}$ to the right of the first pair, as soon as this is received from the client, represented by the action Login to the left of the same pair. Then, the orchestrating actions $\langle \overline{\texttt{Wrong}}, \texttt{Wrong} \rangle$, $\langle \overline{\texttt{Overload}}, \texttt{Overload} \rangle$ and $\langle \overline{\texttt{Ok}}, \texttt{Ok} \rangle$, and the use of the $\vee$ operator, express that, in case BallotOrch gets a message Wrong, Overload or Ok from the server, this message is immediately passed to the client (and the orchestration starts again in case of Wrong or Overload).

In case the message Ok is received, the subsequent orchestrating actions begin by $\langle \varepsilon, \texttt{Id} \rangle . \langle \texttt{VoteB}, \varepsilon \rangle$. The symbol ε represents a *no-action* by the client and by the server respectively, and it has the effect of buffering the other action in the pair. Therefore the message Id from the server is kept in a buffer since the no-action symbol ε to the left of the first orchestrating action replaces the expected $\overline{\texttt{Id}}$. Simlarly the message Vb1 is also kept in the buffer. Only after the reception of the message VoteB, which is immediately passed to the server, the message Vb1 is delivered to the server, and the orchestration stops. The message Id, instead, is never delivered.

The presence of an orchestrator hence allows for both asynchronous interactions and the possibility of disregarding messages. A natural restriction is imposed on orchestrators in [14]: an orchestrator cannot send a message if this has not been previously received. In fact, in the correct orchestrator above, $\langle \texttt{VoteB}, \varepsilon \rangle$ comes before $\langle \varepsilon, \overline{\texttt{VoteB}} \rangle$. This implies that also in Padovani's setting it is not possible to disregard input actions.

The generality of Padovani's notion of orchestrated compliance is paid in terms of a more complex LTS formalizing client/server interaction, which depends on an orchestrator f, that we denote by $\dashv^{\mathsf{P}}_{f}$. In [14] the relation $\rho \dashv^{\mathsf{P}} \sigma$ holds whenever there exists an orchestrator f such that $\rho \dashv^{\mathsf{P}}_{f} \sigma$.

To save decidability of the relevant properties, any correct orchestrator must be of *finite* rank, where the rank of an orchestrator f is the bound of its buffering capability. To make this explicit the notation $\rho \dashv^{\mathsf{P}}_{k} \sigma$ is used whenever there exists an orchestrator f of rank k such that $\rho \dashv^{\mathsf{P}}_{f} \sigma$.

In [14] the sub-behaviour relation induced by orchestrated compliance is defined by:

$$\sigma \preceq^{\mathsf{P}} \sigma' \;\Leftrightarrow\; \forall \rho . [\, \rho \dashv \sigma \Rightarrow \exists f . \, \rho \dashv^{\mathsf{P}}_{f} \sigma'].$$

Notice that the relation $\dashv$ in the antecedent of the implication is just the usual strong compliance. In the same work the relation $\preceq^{\mathsf{P}}$ is proved to be decidable. Moreover the orchestrator f in the definition can be inferred from σ and σ' and it is the same for any possible client ρ.

From what said up to now Padovani orchestrated-compliance relation seems to include ours, since the possibility of *skipping* output actions can be mimicked by orchestrators that keep messages indefinitely inside their buffers, without ever delivering them.

However, apart from the restriction to session behaviours, the two compliance relations are actually *incomparable* because of the finiteness of the ranks of correct orchestrators and of the possibility in our setting to discard infinitely many (non consecutive) output actions from the server side. A counterexample to the inclusion of $\dashv^{\mathsf{skp}}$ in $\dashv^{\mathsf{P}}$ can be obtained by slightly modifying the example used before. Let us consider the ballot service with the extra output action $\overline{\texttt{InfoW}}$, representing some informations about why a login has not been accepted:

$$\begin{array}{l} \mathsf{BallotServiceBehP2} \triangleq \\ \qquad \mathsf{rec}\, x.\, \texttt{Login}.(\overline{\texttt{Wrong}}.\overline{\texttt{InfoW}}.x \oplus \overline{\texttt{Overload}}.x \oplus \overline{\texttt{Ok}}.\overline{\texttt{Id}}.(\;\; \texttt{VoteA}.(\texttt{Va1} + \texttt{Va2}) \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad + \texttt{VoteB}.(\texttt{Vb1} + \texttt{Vb2})\,)\,) \end{array}$$

Consider now the behaviour of a possible voter who can indefinitely try to log-in until (if ever) the login is accepted. This voter is not interested about why a login has not been accepted, nor it is interested in

getting the transaction identifier. Also it does not wish to express a vote for the vice-candidates:
$\mathsf{VoterBehP2} = \mathsf{rec}\,x.\, \overline{\mathtt{Login}}.(\mathtt{Wrong}.\, x + \mathtt{Overload}.\, x + \mathtt{Ok}.\overline{\mathtt{VoteB}})$.
Then we have that $\mathsf{VoterBehP2} \not\dashv^{\mathsf{P}} \mathsf{BallotServiceBehP2}$, that is $\mathsf{VoterBehP2}$ is not compliant with the server $\mathsf{BallotServiceBehP2}$ according to the Padovani's orchestrated compliance. In fact the voter could keep on sending an incorrect login indefinitely, but no correct orchestrator is allowed to buffer an unbounded number of messages, like the $\mathtt{InfoW}$ ones. As a matter of fact, the actual interaction between $\mathsf{VoterBehP2}$ and $\mathsf{BallotServiceBehP2}$ should be carried on in Padovani's setting through the use of an orchestrator like the following one:

$$\begin{array}{lll} \mathsf{BallotOrchP2} \triangleq \mathsf{rec}\,x.\, \langle \mathtt{Login}, \overline{\mathtt{Login}}\rangle.(& & \langle \overline{\mathtt{Wrong}}, \mathtt{Wrong}\rangle.\langle \varepsilon, \mathtt{InfoW}\rangle.x \\ & \vee & \langle \overline{\mathtt{Overload}}, \mathtt{Overload}\rangle.\, x \\ & \vee & \langle \overline{\mathtt{Ok}}, \mathtt{Ok}\rangle.\langle \varepsilon, \mathtt{Id}\rangle.\langle \mathtt{VoteB}, \overline{\mathtt{VoteB}}\rangle\ ) \end{array}$$

which should be able to buffer an unbounded number of $\mathtt{InfoW}$ messages corresponding to the output actions $\overline{\mathtt{InfoW}}$ on the server side. This implies that $\mathsf{BallotOrchP2}$ is not of finite rank and hence it is not correct.

The definition of $\mathtt{skp}$-compliance allows to diregard infinitely many output actions from the server, provided that they are not all consecutive. In particular $\mathsf{VoterBehP2} \dashv^{\mathsf{skp}} \mathsf{BallotServiceBehP2}$. So, formally we get:

Proposition 5.1 *Let $\dashv^{\mathsf{P}}$ be Padovani's weak k-compliance restricted to session behaviours. Then, for any k, we have:*

$$\dashv^{\mathsf{skp}} \;\not\subseteq\; \dashv^{\mathsf{P}}_k$$

The inclusion does hold, instead, if we consider only *finite* behaviours (see Definition 2.4):

Proposition 5.2 *For any pair of* finite *session behaviours ρ, σ, there exists a $k \geq 0$ such that*

$$\rho \dashv^{\mathsf{skp}} \sigma \;\Rightarrow\; \rho \dashv^{\mathsf{P}}_k \sigma$$

In a sense, we think that the $\mathtt{skp}$-compliance relation we investigate in the present paper is the minimal weakening of the standard notion of compliance not requiring the introduction of orchestrators.

6 Conclusion and future work

In the setting of session-behaviors we have relaxed the synchronization rules by allowing output actions on the server side to be skipped by a client that cannot immediately synchronize with them. This gives rise to a weaker notion of compliance, called $\mathtt{skp}$-compliance, and consequently to a new concept of sub-behaviour among servers. We have proved that $\mathtt{skp}$-compliance is still decidable, by exhibiting a derivation system which is sound and complete w.r.t. the new compliance relation, and which is algorithmic, namely it implicitly describes an algorithm to decide $\mathtt{skp}$-compliance. Further we have shown that the duals-as-minima property is preserved in the new setting, which implies decidability of the induced sub-behaviour relation.

In the Introduction we have justified the loosening of compliance by means of examples. Another contexts in which discarding some actions during client/server interaction seems a desirable feature worth to be investigated is that of reversible computations. In particular when the client (or server) of an interaction can roll-back to a previously encountered checkpoint (so forcing a roll-back on the server (client) side). Then the notion of compliance should be strengthened to guarantee that client's requests keep on being satisfied even in case, for any reason, client and server perform a roll-back, as formalized and investigated in [3]. It is not difficult to envisage a situation where the interaction partners could roll-back in two states that would be compliant but for the presence of an output that should have been already sent and received before the roll-back took place. It is reasonable to let the two partners be compliant, since that particular output action could be safely discarded.

Acknowledgements. The authors wish to thank Mariangiola Dezani for her steady and valuable support. Our gratitude also to the anonymous referees that helped us to improve the paper.

References

[1] Eric Badouel & Philippe Darondeau (1991): *On Guarded Recursion*. *Theor. Comput. Sci.* 82(2), pp. 403–408, doi:10.1016/0304-3975(91)90231-P.

[2] Franco Barbanera & Ugo de'Liguoro (2010): *Two notions of sub-behaviour for session-based client/server systems*. In: *Proceedings of PPDP'10*, ACM, pp. 155–164 doi:10.1145/1836089.1836109.

[3] Franco Barbanera, Mariangiola Dezani & Ugo de' Liguoro (2014): *Compliance for reversible client/server interactions*. In: *Proceedings of BEAT 2014*, EPTCS. To appear.

[4] Franco Barbanera & Ugo de' Liguoro (2013): *Sub-behaviour relations for session-based client/server systems*. *MSCS*. To appear.

[5] Giovanni Bernardi & Matthew Hennessy (2013): *Modelling session types using contracts*. *To appear in Mathematical Structures in Computer Science*.

[6] S. Carpineti, G. Castagna, C. Laneve & L. Padovani (2006): *A formal account of contracts for Web Services*. In: *WS-FM, 3rd Int. Workshop on Web Services and Formal Methods*, *LNCS* 4184, Springer, pp. 148–162, doi:10.1007/11841197_10.

[7] G. Castagna, N. Gesbert & L. Padovani (2008): *A Theory of Contracts for Web Services*. In: *POPL '08, 35th ACM Symposium on Principles of Programming Languages*, doi:10.1145/1328438.1328471.

[8] G. Castagna, N. Gesbert & L. Padovani (2009): *Contracts for mobile processes*. In: *Proceedings of the 20th International Conference on Concurrency Theory (CONCUR'09)*, *LNCS* 5710, Springer, pp. 211–228, doi:10.1145/1538917.1538920.

[9] Giuseppe Castagna, Nils Gesbert & Luca Padovani (2009): *A theory of contracts for Web services*. *ACM Trans. Program. Lang. Syst.* 31(5), pp. 19:1–19:61, doi:10.1145/1538917.1538920.

[10] Simon Gay & Malcolm Hole (2005): *Subtyping for Session Types in the Pi-Calculus*. *Acta Informatica* 42(2/3), pp. 191–225, doi:10.1007/s00236-005-0177-z.

[11] Kohei Honda, Vasco T. Vasconcelos & Makoto Kubo (1998): *Language Primitives and Type Disciplines for Structured Communication-based Programming*. In: *ESOP'98*, *LNCS* 1381, Springer-Verlag, pp. 22–138, doi:10.1007/BFb0053567.

[12] Cosimo Laneve & Luca Padovani (2007): *The Must Preorder Revisited: An Algebraic Theory for Web Services Contracts*. In: *CONCUR'07*, *LNCS* 4703, Springer-Verlag, pp. 212–225, doi:10.1007/978-3-540-74407-8_15.

[13] Rocco De Nicola & Matthew Hennessy (1983): *Testing Equivalence for Processes*. In: *ICALP*, *LNCS* 154, Springer, pp. 548–560, doi:10.1007/BFb0036936.

[14] Luca Padovani (2010): *Contract-Based Discovery of Web Services Modulo Simple Orchestrators*. *Theoretical Computer Science* 411, pp. 3328–3347, doi:10.1016/j.tcs.2010.05.002.

[15] Benjamin C. Pierce & Davide Sangiorgi (1996): *Typing and Subtyping for Mobile Processes*. *Mathematical Structures in Computer Science* 6, No. 5.

[16] Davide Sangiorgi (2009): *On the origins of bisimulation and coinduction*. *ACM Trans. Program. Lang. Syst.* 31(4), doi:10.1145/1516507.1516510.

[17] Alfred Tarski (1955): *A lattice-theoretical fixpoint theorem and its applications*. *Pac. J. Math.* 5, p. 285309, doi:10.2140/pjm.1955.5.285.