Journal of Logical and Algebraic Methods in Programming 86 (2017) 30–76

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

Orchestrated session compliance

Franco Barbanera [a,*], Steffen van Bakel [b], Ugo de'Liguoro [c]

[a] Dipartimento di Matematica e Informatica, Università degli Studi di Catania, Viale A. Doria 6, 95125 Catania, Italy
[b] Department of Computing, Imperial College London, 180 Queen's Gate, London SW7 2BZ, UK
[c] Dipartimento di Informatica, Università di Torino, Corso Svizzera 185, 10149 Torino, Italy

ARTICLE INFO

Article history:
Received 15 November 2015
Received in revised form 1 August 2016
Accepted 5 August 2016
Available online 2 September 2016

Keywords:
Compliance
Session contracts
Orchestration
Subcontract

ABSTRACT

We investigate the notion of orchestrated compliance for client/server interactions in the context of *session contracts*. The orchestrators we study have unbounded buffering capabilities and, besides never sending messages which have not been received, are such that any message from the client is eventually delivered by the orchestrator to the server. Moreover, no infinite interaction can consist definitely of messages from the server which are kept by the orchestrator. The subcontract relation induced by this new notion of compliance is also investigated.

1. Introduction

Session types and contracts are two formalisms used to study client/server protocols. Session types have been introduced in [1] as a tool for statically checking safe message exchanges through channels. Contracts as proposed in [2–4] are a subset of CCS [5] without the silent action τ that address the problem of abstractly describing behavioural properties of systems by means of process algebra. In between these two formalisms lie *session contracts*[1] as introduced in [6–9], a formalism interpreting session types into a subset of contracts. Session contracts can be seen as binary session types without value or channel passing (that is, without the so-called "session delegation").

In the theory of contracts, as well as in the formalism of session contracts, the notion of *compliance* plays a central role. A client ρ is called *strongly compliant* with a server σ (written $\rho \dashv \sigma$) whenever *all* of its *requests* are satisfied by the server without ever blocking in any infinite interaction. Now it might be the case that client satisfaction fails just because of a difference in the order in which the partners exchange information, or because one of them provides some extra unneeded information.

Consider the example of a meteorological data processing system (MDPS) that is permanently connected to a weather station to which it sends, according to its processing needs, requests for particular data. For the sake of simplicity, we consider just two particular requests, namely for *temperature* and *humidity*. After sending the requests, the MDPS expects to receive the data in the order they were sent. In the session-contract formalism, the interface for this simplified MDPS can be stated as follows:

$$\mathsf{MDPS} = \mathsf{rec}\, x.\overline{\mathtt{tempReq}}.\overline{\mathtt{humReq}}.\mathtt{temperature}.\mathtt{humidity}.x$$

* Corresponding author.
E-mail addresses: barba@dmi.unict.it (F. Barbanera), svanbakel@imperial.ac.uk (S. van Bakel), ugo.deliguoro@unito.it (U. de'Liguoro).

[1] They were dubbed *session behaviours* in [6,7]. For sake of uniformity and since *session contract* expresses their properties more accurately, we adhere here to this name.

http://dx.doi.org/10.1016/j.jlamp.2016.08.002
2352-2208/

(Here, as in CCS, a symbol like 'a' stands for on input action, whereas '$\overline{a}$' denotes the corresponding output.) We assume a weather station to be able to send back the asked-for information in the order decided by its sensors, interspersed with information about *wind speed*:

$$\mathsf{WeatherStation} = \mathsf{rec}\,x.\,\mathtt{tempReq}.\mathtt{humReq}.(\overline{\mathtt{temperature}}.\overline{\mathtt{humidity}}.\overline{\mathtt{wind}}.x \oplus \overline{\mathtt{humidity}}.\overline{\mathtt{temperature}}.\overline{\mathtt{wind}}.x)$$

With respect to strong compliance, we have that $\mathsf{MDPS} \not\dashv \mathsf{WeatherStation}$, since the client MDPS has no input action for the wind data, and also because it might happen that temperature and humidity data are delivered in a different order than expected by the MDPS. A natural solution is to devise a process that acts as a mediator (here called *orchestrator*) between the client and the server, coordinating them in a centralised way in order to make them compliant. The notion of orchestration has been introduced in the setting of web-service interaction, in particular for business processes:

> "*Orchestration*: Refers to an executable business process that may interact with both internal and external web services. Orchestration describes how web services can interact at the message level, including the business logic and execution order of the interactions." [10]

In the context of the theory of contracts, this solution has been formalised and investigated by Padovani in [11], where orchestrators are processes that cannot affect the internal decisions of the client nor of the server, but can affect the way their synchronisation is carried out.

The communicating actions performed by an orchestrator have the following forms:

$\langle a, \overline{a}\rangle$ (resp. $\langle \overline{a}, a\rangle$): the orchestrator gets a from the client (resp. server) and immediately delivers it to the server (resp. client) in a synchronous way.
$\langle a, \varepsilon\rangle$ (resp. $\langle \varepsilon, a\rangle$): the orchestrator gets a from the client (resp. server) and stores it in a buffer.
$\langle \overline{a}, \varepsilon\rangle$ (resp. $\langle \varepsilon, \overline{a}\rangle$): the orchestrator takes a from the buffer and sends it to the client (resp. server).

So an orchestrator enabling compliance for our example is

$$\mathsf{f} = \mathsf{rec}\,x.\langle \mathtt{tR}, \overline{\mathtt{tR}}\rangle.\langle \mathtt{hR}, \overline{\mathtt{hR}}\rangle.(\langle \overline{\mathtt{t}}, \mathtt{t}\rangle.\langle \overline{\mathtt{h}}, \mathtt{h}\rangle.\langle \varepsilon, \mathtt{w}\rangle.x \vee \langle \varepsilon, \mathtt{h}\rangle.\langle \overline{\mathtt{t}}, \mathtt{t}\rangle.\langle \overline{\mathtt{h}}, \varepsilon\rangle.\langle \varepsilon, \mathtt{w}\rangle.x) \tag{1}$$

where $\mathtt{tR}, \mathtt{hR}, \mathtt{t}, \mathtt{h}$, and $\mathtt{w}$ stand for $\mathtt{tempReq}, \mathtt{humReq}, \mathtt{temperature}, \mathtt{humidity}$, and $\mathtt{wind}$, respectively. The orchestrator f rearranges the order of messages when necessary, and *retains* the wind information, not needed by MDPS.

The orchestrator f is not a valid orchestrator according to [11]: indeed the $\mathtt{wind}$ information is discarded by never delivering it to the client, so that the buffer corresponding to f must be unbounded. Unbounded buffers are not allowed in [11], where boundedness is essential to guarantee decidability of the compliance relation and the possibility of synthesising orchestrators.

We investigate the notion of orchestrated compliance in the setting of session contracts, even in presence of unbounded buffering capabilities of orchestrators. In this system it will be possible to prove that

$$\mathsf{MDPS} \dashv\!\vdash_{\mathsf{f}} \mathsf{WeatherStation}$$

i.e. that MDPS and WeatherStation are compliant when their interaction is mediated as f (also denoted by $\mathsf{f} : \mathsf{MDPS} \dashv\!\vdash \mathsf{WeatherStation}$).

In a two-party session-based interaction, the choice among several continuations always depends on exactly one of the two actors. Hence *session orchestrators* are designed so that internal decisions of the parties are unaffected by the orchestration and any non-deterministic choice depends solely on the partners. This is obtained by restricting the syntax for orchestrators such that for instance orchestrators like $\langle \varepsilon, a\rangle.\mathsf{f}_1 \vee \langle b, \varepsilon\rangle.\mathsf{f}_2$ are not allowed. In fact, in the latter orchestrator, the choice of receiving an input from the client or from the server would not depend on the partners.

In our system we aim at ruling out certain fake complying interactions, like between the client $\overline{a}.\overline{b}$ and the server a through the orchestrator $\langle a, \overline{a}\rangle.\langle b, \varepsilon\rangle$. In this case the client gets the illusion that all its requests are satisfied, whereas its output $\overline{b}$ never reaches the server, being indefinitely kept inside the orchestrator's buffer. While in the contract setting of [11] such interactions are allowed and these parties are compliant, in our context we forbid an orchestrators to behave like $\langle a, \overline{a}\rangle.\langle b, \varepsilon\rangle$ which never deliver a message from the client to the server.

Another sort of fake compliance we will prevent is that between a client like $\overline{a}.\overline{b}$ and a server like $a.\mathsf{rec}\,x.\overline{c}$ by means of the orchestrator $\langle a, \overline{a}\rangle.\mathsf{rec}\,x.\langle \varepsilon, c\rangle$. This is because this interaction would go on indefinitely even if after the satisfaction of the first client's request, the orchestrator indefinitely stores all the server's messages. Such unwanted behaviour of an orchestrator is automatically ruled out in [11] by the boundedness constraint on buffers, while we admit unbounded buffers. On the other hand, as done also in [11], we rule out orchestrators sending messages that they have never received.

All the properties characterising well-behaved orchestrators, which we shortly dub *respectfulness*, will be shown to be decidable. Turning to the meteorological example, it turns out that the orchestrator f for MDPS and WeatherStation is respectful. As a matter of fact, respectfulness is introduced as a property of *traces* of orchestrated interactions; an orchestrator is respectful if all its traces, that include those of the system of the parties and the orchestrator, are such. The class of respectful orchestrators, however, does not admit a simple, 'inductive' characterisation. Nonetheless, whether an orchestrator f is respectful is decidable, as well as the relation $\dashv\!\vdash_f$.

The relation of orchestrated compliance through a (respectful) orchestrator f does naturally induce an *orchestrated subcontract relation* on servers, that we dub $\overset{\circ}{\preccurlyeq}_f$. The notion of compliance naturally induces a substitutability relation. In any theory of subcontracts, given a compliance relation, the relevance of the induced subcontract relation is due to its being the formal counterpart of the abstract notion of substitutability. This is quite relevant for practical applications. The subcontract relation can be safely used to update the services in a repository in order to widen the range of the clients which can use it. Also the implementation of contract-based query engines can exploit properties of this relation (see [11] for a detailed discussion).

We prove a number of properties of the relation $\overset{\circ}{\preccurlyeq}_f$, including decidability, dual-precedes-all-servers, a parametrised form of transitivity, and what we call substitutability embodiment. The dual-precedes-all-servers property states that, given a contract ρ and an orchestrator f, the contract $\overline{\rho}$ precedes (with respect to the relation $\overset{\circ}{\preccurlyeq}_f$) all the servers of ρ orchestrated by f, where $\overline{\rho}$ is the standard dual of ρ (i.e. it is obtained by exchanging input and output actions, and by exchanging $+$ and $\oplus$). Since our notion of subcontract is parametrised by orchestrators, the form of transitivity we prove consists in the statement

$$\rho \overset{\circ}{\preccurlyeq}_f \delta \ \&\ \delta \overset{\circ}{\preccurlyeq}_g \sigma \implies \rho \overset{\circ}{\preccurlyeq}_{f\bullet g} \sigma,$$

where $\cdot\bullet\cdot$ is an appropriate (computable) orchestrator composition operation.

The above orchestrator composition operation is relevant to guarantee also that the relation $\overset{\circ}{\preccurlyeq}_f$ actually formalises a general notion of substitutability. In fact, the substitutability embodiment property that we prove consists in the statement

$$\sigma \overset{\circ}{\preccurlyeq}_g \sigma' \ \&\ f : \rho \dashv\!\vdash \sigma \implies f\bullet g : \overline{\rho} \dashv\!\vdash \sigma' \tag{2}$$

When $\sigma \overset{\circ}{\preccurlyeq}_g \sigma'$, the server σ can be safely replaced by the server σ' since any client interacting with σ using an orchestrator f can safely interact with σ' after the new orchestrator $f\bullet g$ has been computed.

It is worth remarking that statement (2) does not immediately descend from the definition of our subcontract relation, differently from what is the case for relations induced by other notions of compliance.

Let us now consider our previous example, and assume to have a new weather station whose behaviour is described by the following contract.

$$\begin{aligned}&\mathsf{NewWeatherStation} = \\ &\quad \mathsf{rec}\,x.(\mathtt{humReq}.\mathtt{tempReq}.\overline{\mathtt{wind}}.\overline{\mathtt{humidity}}.\overline{\mathtt{temperature}}.\overline{\mathtt{pressure}}.x \\ &\qquad + \\ &\qquad \mathtt{reset}.x)\end{aligned}$$

As we shall see, it is possible to formally prove that

$$\mathsf{WeatherStation} \overset{\circ}{\preccurlyeq}_{\mathsf{g}} \mathsf{NewWeatherStation} \tag{3}$$

where $\mathsf{g} = \mathsf{rec}\,x.\langle \mathtt{tR}, \varepsilon\rangle.\langle \mathtt{hR}, \overline{\mathtt{hR}}\rangle.\langle \varepsilon, \overline{\mathtt{tR}}\rangle.\langle \varepsilon, \mathtt{w}\rangle.\langle \overline{\mathtt{h}}, \mathtt{h}\rangle.\langle \overline{\mathtt{t}}, \mathtt{t}\rangle.\langle \overline{\mathtt{w}}, \varepsilon\rangle.\langle \varepsilon, \mathtt{p}\rangle.x$.

Now property (2) guarantees that in our "meteorological distributed system", the server WeatherStation can be safely replaced by the NewWeatherStation. In fact, if a client had to use an orchestrator to interact with WeatherStation, we could compose such an orchestrator with g in order to let the client safely interact also with the NewWeatherStation. Take the client MDPS and the orchestrator f described in (1). We know that f enables MDPS to use the server WeatherStation, that is $\mathsf{f} : \mathsf{MDPS} \dashv\!\vdash \mathsf{WeatherStation}$.

The computation of $\mathsf{f}\bullet\mathsf{g}$, as defined in the paper, returns the following result.

$$\mathsf{f}\bullet\mathsf{g} \ = \ \mathsf{rec}\,x.\langle \mathtt{tR}, \varepsilon\rangle.\langle \mathtt{hR}, \overline{\mathtt{hR}}\rangle.\langle \varepsilon, \overline{\mathtt{tR}}\rangle.\langle \varepsilon, \mathtt{w}\rangle.\langle \varepsilon, \mathtt{h}\rangle.\langle \overline{\mathtt{t}}, \mathtt{t}\rangle.\langle \overline{\mathtt{h}}, \varepsilon\rangle.\langle \varepsilon, \mathtt{p}\rangle.x$$

It can now be checked that

$$\mathsf{f}\bullet\mathsf{g} : \mathsf{MDPS} \dashv\!\vdash \mathsf{NewWeatherStation}$$

All the properties of the relation $\rho \overset{\circ}{\preccurlyeq}_f \delta$ will be obtained as consequences of what we call the Main Property:

$$\rho \dashv\!\vdash_f \sigma \ \&\ \overline{\sigma} \dashv\!\vdash_g \sigma' \implies \rho \dashv\!\vdash_{(f\bullet g)} \sigma'$$

The proof of the Main Property is relatively simple when orchestration is not considered. In our context, instead, its proof turns out to be rather complex. In fact, one has to show not only that $f\bullet g$ correctly orchestrates the interactions between ρ and σ', but also that it is a respectful orchestrator when f and g are, which is actually the case in the hypothesis of the Main Property.

A natural generalisation of the relation $\overset{\circ}{\preccurlyeq}_f$ is $\sigma \overset{\circ}{\preccurlyeq} \sigma'$, which holds if there exists a respectful f such that $\sigma \overset{\circ}{\preccurlyeq}_f \sigma'$. This subcontract relation can be obtained via a compliance relation $\dashv\!\vdash$ which is the union of all the $\dashv\!\vdash_f$ relations. It is possible to show that the relation $\overset{\circ}{\preccurlyeq}$ is decidable if and only if $\dashv\!\vdash$ is such. To face the issue, in [12] we have proposed an algorithm aiming at synthesising a respectful orchestrator f, if any such an orchestrator exists, such that $\rho \dashv\!\vdash_f \sigma$, and failing otherwise. Although that algorithm is sound, it cannot be proved to be complete: see Remark 6.13 for more details.

In [13] a notion of compliance weaker than strong compliance was introduced: a client is compliant with a server also when output from the server which does not correspond to any input request by the client must be discarded ('skipped') until the satisfaction state is reached. This weaker compliance relation was dubbed `skp`-compliance, and denoted by the symbol $\dashv^{\mathtt{skp}}$. In [13] a discussion can be found about why it would be unreasonable to skip an input action (roughly since the continuation of a computation could depend on it) or an output action from the client (which, according to the *client-biased* viewpoint of compliance in general, are 'requests' that must be satisfied).

For such a relation, the MDPS client defined in the introduction is compliant with the following version of the server WeatherStation

$$\mathsf{WeatherStationII} = \mathsf{rec}\, x.\, \mathtt{tempReq}.\mathtt{humReq}.\overline{\mathtt{temperature}}.\overline{\mathtt{humidity}}.\overline{\mathtt{wind}}.x$$

which, after satisfying the requests of the client in the given order, sends to it the useless output $\overline{\mathtt{wind}}$. That is, MDPS $\dashv^{\mathtt{skp}}$ WeatherStationII.

As shown in [13], the relation $\dashv^{\mathtt{skp}}$ is decidable and enjoys all the desirable properties of strong compliance $\dashv$. Hence in a service-oriented programming scenario it can be used to widen the spectrum of the possible servers one can use.

Whenever a server σ is found for a client ρ such that $\rho \dashv^{\mathtt{skp}} \sigma$, the full handshaking communication mechanism of ρ should be modified in order to prevent the client from blocking in case of an unneeded output from σ, which should instead be discarded. Changing the communication mechanism of a client according to the sort of compliance relation that holds with its server could result in unfeasible overhead. In case a client ρ were found to be `skp`-compliant with a server, it would be definitely more feasible just to have their communications mediated by an orchestrator, whose task would be just to discard the unneeded outputs from the server. In the orchestrator formalism this corresponds to keeping them indefinitely in an unbounded buffer. We shall then prove the relation of `skp`-compliance to be equivalent to a restricted version of our orchestrated compliance, that we dub `skp-Orch`-compliance. Moreover, for `skp-Orch`-compliance, it is possible to define a correct and complete synthesis algorithm finding a suitable orchestrator for a client ρ and a server σ in case they are `skp-Orch`-compliant.

The present paper is a revised and extended version of [12], where the proofs concerning the results for the relations $\rho \dashv\!\vdash_f \sigma$ were sketched, and where the relations $\overset{\circ}{\preccurlyeq}_f$ were not taken into account. Also, the equivalence of $\dashv^{\mathtt{skp}}$ with the relation of `skp-Orch`-compliance was not shown in [12]. As mentioned previously, in [12] a synthesis algorithm for orchestrators for the relation $\dashv\!\vdash$ was claimed to be complete; but actually only its soundness can be proved, as will be discussed in this paper.

Overview of the paper In Section 2 we will introduce and formalise the notion of orchestrator, together with the relation $f : \rho \dashv\!\vdash^{\mathsf{d}} \sigma$ on which the relation of orchestrated compliance is based. A formal system for the relation $f : \rho \dashv\!\vdash^{\mathsf{d}} \sigma$ will be presented in Section 3 where it will also be proved to be sound and complete; the most technical part of the proofs are deferred to Appendix A in order to improve readability. Out of the proof of completeness of the formal system for $f : \rho \dashv\!\vdash^{\mathsf{d}} \sigma$, we shall get the proof of decidability for such a relation. This proof is based also on another, equivalent, formal system. It will be formalised and proven equivalent in Appendix C. In Section 4 we shall formalise the notion of respectfulness for orchestrators. This will enable us to define the relation $f : \rho \dashv\!\vdash \sigma$ in terms of $f : \rho \dashv\!\vdash^{\mathsf{d}} \sigma$ and of the respectfulness property for orchestrators. The decidability of the respectfulness property will be proved in Appendix B for readability sake, since it depends on a collection of rather technical definitions. We are hence able to get decidability for $f : \rho \dashv\!\vdash \sigma$ as a corollary of decidability of the formal system for $f : \rho \dashv\!\vdash^{\mathsf{d}} \sigma$ and decidability of the respectfulness property. The orchestrated subcontract relation $\overset{\circ}{\preccurlyeq}_f$ will be formalised in Section 5 and several results about it will be proved. Also $\overset{\circ}{\preccurlyeq}$ will be investigated in Section 5. Most of the results of this section will be obtained out of what we call the Main Property; since its proof rather is complex, we shall postpone it to Appendix D. Section 6 will deal with the proof of equivalence for the relations of `skp`-compliance and `skp-Orch`-compliance. In that section a synthesis algorithm will be provided for orchestrators for the relation of `skp-Orch`-compliance. The algorithm will be proved correct and complete in Appendix E. Conclusions and related and future work will be the topic of Section 7.

2. Session contracts, orchestrators and disrespectful orchestrated compliance

Session contracts are a restriction of *contracts* [3,4]. They are designed to be in one-to-one correspondence to session types [1] without delegation (in [6,7] a version with delegation was investigated). The restriction consists in constraining internal and external choices in a way that limits the non-determinism to (internal) output selection.

σ, ρ	$::= \mathbf{1}$	success
	$\mid a_1.\sigma_1 + \cdots + a_n.\sigma_n$	external choice
	$\mid \overline{a}_1.\sigma_1 \oplus \cdots \oplus \overline{a}_n.\sigma_n$	internal choice
	$\mid x$	variable
	$\mid \mathsf{rec}\, x.\sigma$	recursion

Fig. 1. The grammar of raw session-contracts.

Definition 2.1 *(Session contracts).*

i) Let $\mathcal{N}$ be a countable set of symbols and $\overline{\mathcal{N}} = \{\overline{a} \mid a \in \mathcal{N}\}$. The set RSC of *raw session contracts* is defined by the grammar in Fig. 1, where:
- for external and internal choices, $n \geq 1$, and $a_i \in \mathcal{N}$ (hence $\overline{a}_i \in \overline{\mathcal{N}}$) for all $1 \leq i \leq n$;
- the variable x is a *session-contract variable* out of a denumerable set; we consider occurrences of x in σ *bound* in $\mathsf{rec}\, x.\sigma$. An occurrence of x in σ is *free* if it is not bound, and we write $\mathrm{FV}(\sigma)$ for the set of free variables in σ. σ is said to be *closed* whenever $\mathrm{FV}(\sigma) = \emptyset$.

Act $= \mathcal{N} \cup \overline{\mathcal{N}}$ is the set of *actions* and will be ranged over by the metavariables α, α', α_1, α_2, etc. On **Act** the usual involution $(\overline{\cdot})$ is defined, that is such that $\overline{\overline{\alpha}} = \alpha$.

ii) The set SC of **session contracts** is the subset of closed raw session contracts 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_is and the $\overline{a}_i$s are, in both, pairwise distinct; moreover, in $\mathsf{rec}\, x.\sigma$ the expression σ is not a variable.

As usual, 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 $\oplus_{i=1}^{n} \overline{a}_i.\sigma_i$. We also use the notations $\sum_{i\in I} a_i.\sigma_i$ and $\oplus_{i\in I} \overline{a}_i.\sigma_i$ for finite and non-empty I. We take the equi-recursive view of recursion, by equating $\mathsf{rec}\, x.\sigma$ with $\sigma\{x/\mathsf{rec}\, x.\sigma\}$.

The trailing $\mathbf{1}$ is normally omitted: for example, we will write $a + b$ for $a.\mathbf{1} + b.\mathbf{1}$. Session contracts will be considered modulo commutativity of internal and external choices.

A syntactical concept of *duality* on SC is obtained by interchanging a with $\overline{a}$ and $+$ with $\oplus$. Seen that such a notion is formally defined by induction on the structure of (possibly open) expressions, we first consider raw session contracts in RSC. Duality for elements in SC is then obtained by restricting to SC the duality defined for elements of RSC.

Definition 2.2 *(Syntactic duality).*

i) Let σ be a raw behaviour expression, that is $\sigma \in \mathsf{RSC}$. The syntactic dual $\overline{\sigma}$ of σ is defined by the following clauses:

$$\overline{\mathbf{1}} = \mathbf{1} \qquad \overline{x} = x \qquad \overline{\mathsf{rec}\, x.\sigma} = \mathsf{rec}\, x.\overline{\sigma}$$

$$\overline{\textstyle\sum_{i\in I} a_i.\sigma_i} = \bigoplus_{i\in I} \overline{a}_i.\overline{\sigma_i} \qquad \overline{\textstyle\bigoplus_{i\in I} \overline{a}_i.\sigma_i} = \sum_{i\in I} a_i.\overline{\sigma_i}$$

It is immediate to check that

$$(\#) \qquad \overline{\sigma} \in \mathsf{SC} \iff \sigma \in \mathsf{SC}.$$

ii) We define $\overline{(\cdot)} : \mathsf{SC} \to \mathsf{SC}$ as the restriction to SC of the duality function of item (i), using the observation (#).

From now on, in order to avoid too cumbersome definitions, any time an inductive definition on elements of SC is provided, it will be tacitly assumed to be actually the restriction to SC of the corresponding inductive definition on RSC. Definition 2.2 closely mimics the duality operator on session types as defined for example in [14].

The operational semantics of session contracts is given in terms of a labelled transition system (LTS) $\sigma \xrightarrow{\beta} \sigma'$ where $\sigma, \sigma' \in \mathsf{SC}$ and β either belongs to the set of actions **Act** or is an internal action τ.

Definition 2.3 *(LTS for session contracts).* We define the labelled transition system (LTS) $(\mathsf{SC}, \mathbf{Act}, \to)$ by the rules:

$$\overline{a}_1.\sigma_1 \oplus \cdots \oplus \overline{a}_n.\sigma_n \xrightarrow{\tau} \overline{a}_k.\sigma_k \quad (n \geq 2)$$

$$\overline{a}.\sigma \xrightarrow{\overline{a}} \sigma$$

$$a_1.\sigma_1 + \cdots + a_n.\sigma_n \xrightarrow{a_k} \sigma_k \quad (n \geq 1)$$

where $1 \leq k \leq n$. We shall use $\to$ as shorthand for $\xrightarrow{\tau}$ and $\xRightarrow{\alpha}$ for $\to^* \xrightarrow{\alpha} \to^*$ with $\alpha \in \mathbf{Act}$.

Notice that reduction is not defined through contextual rules, so reduction only takes place at the 'top' level. Thereby, it is impossible for $\mathsf{rec}\, x.\sigma$ to unfold more than once without consuming a guard (remember that σ is not a variable): so

recursion is contractive in the usual sense. We will safely assume that no two consecutive rec binders (as in $\mathsf{rec}\,x.\mathsf{rec}\,y.\sigma$) are present in a session contract.

We observe that $\stackrel{\alpha}{\Longrightarrow}$ is well defined, in that if $\sigma \in \mathsf{SC}$ and $\sigma \stackrel{\alpha}{\Longrightarrow} \sigma'$ (or $\sigma \to \sigma'$), then $\sigma' \in \mathsf{SC}$.

Session orchestrators As has been done in [11] in the context of the theory of contracts, we will investigate the notion of compliance when the interaction between a client and a server is mediated by an *orchestrator*. Different from the broad contract setting, the session setting we are in induces some natural restrictions to the syntax of orchestrators, making it safe to have orchestrators with unbounded buffers. Moreover, it is possible to check whether any output from the client is eventually delivered by the orchestrator to the server, as well as whether there might be an infinite interaction which falsely progresses because it is made only of output from the server to the orchestrator (see Section 4).

The set of actions an orchestrator can perform, which we take from [11], have been informally described in the introduction.

Remark 2.4. One could wonder whether just asynchronous orchestration actions could be taken into account, since it seems that any $\langle a, \overline{a}\rangle$ action can be safely mimicked by two asynchronous ones, namely $\langle a, \varepsilon\rangle.\langle \varepsilon, \overline{a}\rangle$ (similarly for $\langle \overline{a}, a\rangle$). As a matter of fact, this is not the case if we wish a general formalism as much as possible, considering also the possibility of distinguishing different occurrences of the same message. Let us consider for instance the following orchestrators

$$g \;=\; \langle a, \varepsilon\rangle.\mathsf{rec}\,x.\langle a, \overline{a}\rangle.x \qquad \text{and} \qquad g' \;=\; \langle a, \varepsilon\rangle.\mathsf{rec}\,x.\langle a, \varepsilon\rangle.\langle \varepsilon, \overline{a}\rangle.x$$

The first message 'a' received by g will be indefinitely kept inside its buffer, whereas all the other messages 'a' will be synchronously delivered to the server.

For what concerns g', instead, if its buffer is FIFO queue, the first message received will be delivered to the server after the arrival of the second message. In particular, all the received 'a'-messages will eventually be delivered to the server. This means that according to the notion of *respectful* orchestrator that we shall formalise in Section 4, g' is a respectful orchestrator, whereas g it is not. The need to let actions $\langle a, \overline{a}\rangle$ be non-equivalent to $\langle a, \varepsilon\rangle.\langle \varepsilon, \overline{a}\rangle$ (and similarly for $\langle \overline{a}, a\rangle$) will be recalled in Remark 4.5.

It can reasonably be argued that orchestrators must not show any internal non-determinism. While considering the *session-based* interactions in our setting, such an assumption should be further strengthened, keeping in mind that, in a session-based client/server interaction, any possible non-determinism is due only to the internal non-determinism of the two partners. We therefore define *session-orchestrators* enforcing this point of view. It follows that the only choice we allow in session orchestrators (represented by '$\vee$' in expressions like $f \vee g$) is an external one, and is necessarily driven by the internal choice of one of the two partners. This implies that the actions immediately exhibited by f and g in an orchestrator like $f \vee g$ must have the same *direction*, i.e. it must belong to just one of the two subsets $\{\langle a, \varepsilon\rangle, \langle a, \overline{a}\rangle \mid a \in \mathcal{N}\}$ or $\{\langle \varepsilon, a\rangle \mid \langle \overline{a}, a\rangle \mid a \in \mathcal{N}\}$. We prevent also the orchestrator's non-determinism due to disjunctions like $\langle a, \varepsilon\rangle.f_1 \vee \langle a, \overline{a}\rangle.f_2$. Besides, orchestration actions of the form $\langle \overline{a}, \varepsilon\rangle$ or $\langle \varepsilon, \overline{a}\rangle$ must be used just as prefixes μ in orchestrators like $\mu.f$.

The rationale underlying the other ruled-out cases is as follows. Orchestrators behaving like $\langle \overline{c}, \varepsilon\rangle.f' \vee \langle \varepsilon, b\rangle.g'$ or $\langle c, \varepsilon\rangle.f' \vee \langle \varepsilon, b\rangle.g'$ are ruled out since they would conflict with the session viewpoint (according to which at any moment the progress of an interaction depends on just one of the two partners). Orchestrators like $\langle \overline{c}, \varepsilon\rangle.f' \vee \langle b, \varepsilon\rangle.g'$, instead, would not take into account the form of session contracts since it is impossible a contract could either send an output or receive an input.

Remark 2.5.[2] Notice that, by the above mentioned restriction we cannot make a client $\rho = \overline{a}.b$ compliant with a server $\sigma = \overline{b}.a$ by means of orchestrators like

$$\begin{array}{l}\langle \varepsilon, b\rangle.\langle a, \varepsilon\rangle.(\langle \overline{b}, \varepsilon\rangle.\langle \varepsilon, \overline{a}\rangle \vee \langle \varepsilon, \overline{a}\rangle.\langle \overline{b}, \varepsilon\rangle) \\ \qquad\qquad\qquad \vee \\ \langle a, \varepsilon\rangle.\langle \varepsilon, b\rangle.(\langle \overline{b}, \varepsilon\rangle.\langle \varepsilon, \overline{a}\rangle \vee \langle \varepsilon, \overline{a}\rangle.\langle \overline{b}, \varepsilon\rangle)\end{array}$$

The compliance of the partners ρ and σ obtained by means of the above orchestrator would correspond to the interaction of two CFSMs by means of FIFO buffers. In our formalism, instead, we have to impose a *session-like* evolution of the system, by choosing beforehand one of the possible traces of actions enabling the evolution of the system, for instance by means of an orchestrator like

$$\langle \varepsilon, b\rangle.\langle a, \varepsilon\rangle.\langle \overline{b}, \varepsilon\rangle.\langle \varepsilon, \overline{a}\rangle$$

So, the loss in generality of our formalism does not disrupt the possibility of ρ and σ be made compliant. Moreover, it enables us to get a more manageable system.

[2] We are indebted to an anonymous referee for this remark.

We now formally define orchestration actions by partitioning them into different syntactic categories.

Definition 2.6 *(Session-orchestration actions).* We define **OrchAct** as the set of *session-orchestration actions* μ described by the following grammar (where $a \in \mathcal{N}$ and $\overline{a} \in \overline{\mathcal{N}}$):

$$\begin{array}{lll} \mu & ::= & \iota_L \mid \iota_R \mid o \\ \iota_L & ::= & \langle a, \varepsilon \rangle \mid \langle a, \overline{a} \rangle \end{array} \qquad \begin{array}{lll} o & ::= & \langle \overline{a}, \varepsilon \rangle \mid \langle \varepsilon, \overline{a} \rangle \\ \iota_R & ::= & \langle \varepsilon, a \rangle \mid \langle \overline{a}, a \rangle \end{array}$$

where $\mu \in \iota_L$ or $\mu \in \iota_R$ if the input part of μ is on the left or on the right hand side respectively. We let $\mu, \mu', \mu_1, \ldots$ range over orchestration actions, and $\boldsymbol{\mu}$ over both finite sequences $\mu_1 \cdots \mu_n$ in $\mathbf{OrchAct}^*$ and infinite sequences $\mu_1 \cdots \mu_n \cdots$ in $\mathbf{OrchAct}^\infty$.

Let the set I be either of the form $\{\langle a_1, \phi_1 \rangle, \ldots, \langle a_n, \phi_n \rangle\} \subseteq \iota_L$ or of the form $\{\langle \phi_1, a_1 \rangle, \ldots, \langle \phi_n, a_n \rangle\} \subseteq \iota_R$, where $\phi_i \in \{\varepsilon\} \cup \overline{\mathcal{N}}$ $(1 \leq i \leq n)$. We say that the set I is *input disjoint* whenever all the a_i are pairwise distinct.

Session orchestrators are now defined as follows.

Definition 2.7 *(Session orchestrators).* The set Orch, ranged over by $f, g, h, \ldots$, of *session orchestrators* (or *orchestrators* for short) is the subset of closed *orchestrator expressions*, the latter being generated by the grammar:

$$\begin{array}{rcll} f, g & ::= & \mathbf{1} & \\ & \mid & \iota_L.f_1 \vee \cdots \vee \iota_L.f_n & (n \geq 1) \\ & \mid & \iota_R.f_1 \vee \cdots \vee \iota_R.f_n & (n \geq 1) \\ & \mid & o.f & \\ & \mid & x & \\ & \mid & \mathsf{rec}\, x.f & \end{array}$$

where recursion is *contractive*, namely the f in $\mathsf{rec}\, x.f$ is not a variable. Moreover, we assume that for any orchestrator $\mu_1.f_1 \vee \cdots \vee \mu_n.f_n$ $(n \geq 1)$, the set $\{\mu_i, \ldots, \mu_n\}$ is *input disjoint*.

The expression $\mathbf{1}$ represents the orchestrator offering no action. $o.f$ offers just the orchestration action of the category o and continues as f, whereas $\iota_L.f_1 \vee \cdots \vee \iota_L.f_n$ and $\iota_R.f_1 \vee \cdots \vee \iota_R.f_n$ offer n (uni-directional) actions of, respectively, the syntactical categories ι_L and ι_R, which are input disjoint. Recursive orchestrators can be expressed by means of the rec binder and recursion variables, in the usual way. As for session contracts, orchestrators are defined to have recursion variables guarded by at least one orchestration action. As for session contracts, unless specified otherwise, we take an equi-recursive point of view of orchestrators, so identifying $\mathsf{rec}\, x.f$ and $f\{x/\mathsf{rec}\, x.f\}$.

We now define the operational semantics of orchestrators as an LTS.

Definition 2.8 *(LTS for orchestrators).* We define the labelled transition system (Orch, **OrchAct**, $\mapsto$) by

$$\frac{}{\mu.f \overset{\mu}{\longmapsto} f} \qquad \frac{f \overset{\mu}{\longmapsto} f'}{f \vee g \overset{\mu}{\longmapsto} f'} \qquad \frac{g \overset{\mu}{\longmapsto} g'}{f \vee g \overset{\mu}{\longmapsto} g'}$$

Given a sequence $\boldsymbol{\mu}$, we write $f \overset{\boldsymbol{\mu}}{\longmapsto}$ whenever $f \overset{\mu_1}{\longmapsto} f_1 \overset{\mu_2}{\longmapsto} \cdots \overset{\mu_n}{\longmapsto} f_n$ if $\boldsymbol{\mu} = \mu_1 \cdots \mu_n \in \mathbf{OrchAct}^*$, or $f \overset{\mu_1}{\longmapsto} \cdots \overset{\mu_n}{\longmapsto} f_n \overset{\mu_{n+1}}{\longmapsto} \cdots$ if $\boldsymbol{\mu} = \mu_1 \cdots \mu_n \cdots \in \mathbf{OrchAct}^\infty$. We write $f \not\mapsto$ if there is no μ such that $f \overset{\mu}{\longmapsto}$.

Definition 2.9 *(Orchestrator traces).* Let $f \in \mathsf{Orch}$.

1. The set $\mathsf{Tr}(f) \subseteq (\mathbf{OrchAct}^* \cup \mathbf{OrchAct}^\infty)$ of *traces of* f is defined by:

$$\mathsf{Tr}(f) = \{\boldsymbol{\mu} \mid f \overset{\boldsymbol{\mu}}{\longmapsto}\}.$$

2. The set $\mathsf{MaxTr}(f) \subseteq (\mathbf{OrchAct}^* \cup \mathbf{OrchAct}^\infty)$ of *maximal traces of* f is defined by:

$$\mathsf{MaxTr}(f) = \{\boldsymbol{\mu} \in \mathsf{Tr}(f) \mid \exists f' \,[\, f \overset{\boldsymbol{\mu}}{\longmapsto} f' \not\mapsto \,] \text{ or } \boldsymbol{\mu} \in \mathbf{OrchAct}^\infty\}$$

As in [11], we define an *orchestrated system* as a triple $\langle \rho, f, \sigma \rangle$ (written $\rho \parallel_f \sigma$) that represents ρ (the client) and σ (the server) interacting with each other under the supervision of f (the orchestrator).

Definition 2.10 *(Operational semantics of orchestrated systems).* The operational semantics of orchestrated systems is defined as follows:

$$\frac{\rho \xrightarrow{\tau} \rho'}{\rho \parallel_f \sigma \xrightarrow{\tau} \rho' \parallel_f \sigma} \qquad \frac{\sigma \xrightarrow{\tau} \sigma'}{\rho \parallel_f \sigma \xrightarrow{\tau} \rho \parallel_f \sigma'} \qquad \frac{\rho \xrightarrow{\alpha} \rho' \quad f \overset{\langle \overline{\alpha}, \alpha \rangle}{\longmapsto} f' \quad \sigma \xrightarrow{\overline{\alpha}} \sigma'}{\rho \parallel_f \sigma \xrightarrow{\langle \overline{\alpha}, \alpha \rangle} \rho' \parallel_{f'} \sigma'}$$

$$\frac{\rho \xrightarrow{\overline{\alpha}} \rho' \quad f \overset{\langle \alpha, \varepsilon \rangle}{\longmapsto} f'}{\rho \parallel_f \sigma \xrightarrow{\langle \alpha, \varepsilon \rangle} \rho' \parallel_{f'} \sigma} \qquad \frac{f \overset{\langle \varepsilon, \alpha \rangle}{\longmapsto} f' \quad \sigma \xrightarrow{\overline{\alpha}} \sigma'}{\rho \parallel_f \sigma \xrightarrow{\langle \varepsilon, \alpha \rangle} \rho \parallel_{f'} \sigma'}$$

We write $\overset{\mu}{\Longrightarrow}$ for $\xrightarrow{\tau}{}^* \circ \xrightarrow{\mu} \circ \xrightarrow{\tau}{}^*$, and $\overset{\boldsymbol{\mu}}{\Longrightarrow}$ for $\overset{\mu_1}{\Longrightarrow} \circ \cdots \circ \overset{\mu_n}{\Longrightarrow}$ (resp. $\overset{\mu_1}{\Longrightarrow} \circ \overset{\mu_2}{\Longrightarrow} \circ \cdots$) if $\boldsymbol{\mu}$ is finite (resp. infinite). The notation $\rho \parallel_f \sigma \longrightarrow$ will be used for either $\rho \parallel_f \sigma \xrightarrow{\tau}$ or $\exists \mu\, [\, \rho \parallel_f \sigma \xrightarrow{\mu}\,]$.

Notice that for the operational semantics of orchestrated systems we have defined labelled reductions instead of a reduction relation (as done in [11]). We label orchestrated-systems' transitions by the orchestration actions which make them possible, since in our setting we need to check for particular conditions of orchestrator buffers after the evolution of an orchestrated system. A buffer can be explicitly coupled with an orchestrator or can be represented implicitly by the actions performed by the orchestrator. The latter is the choice of [11], that we maintain.

Disrespectful orchestrated compliance We now define a notion of compliance which is coarser than expected because of possible unfair behaviour of the orchestrators, which will be refined in Section 4.

Definition 2.11 *(Disrespectful orchestrated compliance).* We define $f : \rho \dashv\!\shortmid^{\mathsf{d}} \sigma$ as follows:

$$f : \rho \dashv\!\shortmid^{\mathsf{d}} \sigma \triangleq \forall \rho', \sigma', f' \, [\, \rho \parallel_f \sigma \overset{\boldsymbol{\mu}}{\Longrightarrow} \rho' \parallel_{f'} \sigma' \not\longrightarrow \implies \rho' = \mathbf{1}\,].$$

Remark 2.12.[3] Notice that if no restriction is imposed on the behaviour of orchestrators, given a client–server pair $\rho \parallel \sigma$, it is always possible to get an orchestrator f_ρ, depending just on the client ρ, such that $f_\rho : \rho \dashv\!\shortmid^{\mathsf{d}} \sigma$. The orchestrator f_ρ can be easily defined on the structure of ρ and produces all the input needed by ρ and keeps inside itself all the outputs coming from ρ, without ever interacting with σ.

Strict orchestrators When investigating the relation of disrespectful orchestrated compliance (and the respectful one as well), we can actually take into account a restricted class of orchestrators.

Definition 2.13 *(Strict orchestrators).* An orchestrator f is said to be ρ-σ *strict*[4] whenever, for any finite $\boldsymbol{\mu}$, $f \overset{\boldsymbol{\mu}}{\longmapsto}$ implies $\rho \parallel_f \sigma \overset{\boldsymbol{\mu}}{\Longrightarrow}$.

Example 2.14. For instance, let us consider $f : \mathsf{rec}\, x\, .\, a.x \dashv\!\shortmid^{\mathsf{d}} \mathsf{rec}\, x\, .\, \overline{a}.\overline{c}.x$. The orchestrator

$$f \;=\; \mathsf{rec}\, x\, .(\langle \overline{a}, a \rangle .x \vee \langle \varepsilon, c \rangle .x)$$

is not strict, since $f \overset{\langle \overline{a}, a \rangle . \langle \overline{a}, a \rangle}{\longmapsto}$, but $\mathsf{rec}\, x\, .\, a.x \parallel_f \mathsf{rec}\, x\, .\, \overline{a}.\overline{c}.x \overset{\langle \overline{a}, a \rangle \langle \overline{a}, a \rangle}{\Longrightarrow\!\!\!\!\!\!/\;\;}$. A strict orchestrator is instead $f^{\mathbf{s}} = \mathsf{rec}\, x\, .\langle \overline{a}, a \rangle . \langle \varepsilon, c \rangle .x$.

We manage to obtain $f^{\mathbf{s}}$ out of the non-strict f via an algorithm, that consists of a recursive pruning of f, which is a regular tree. Since $\vee$ behaves as the external choice in CCS, it is observationally commutative and associative, so that we can freely abbreviate $\mu_1.f_1 \vee \cdots \vee \mu_n.f_n$ $(n \geq 1)$ by $\bigvee_{i \in I} \mu_i.f_i$ where $I = \{1, \ldots, n\}$. This notation is overloaded in $\bigvee\{\mu_1.f_1, \ldots, \mu_1.f_n\}$; also we identify $\bigvee_{i \in \emptyset} \mu_i.f_i$ with both $\bigvee \emptyset$ and $\mathbf{1}$.

Proposition 2.15. *Let $f : \rho \dashv\!\shortmid^{\mathsf{d}} \sigma$. Then it is possible to compute a ρ-σ-strict orchestrator $f^{\mathbf{s}}$ such that $f^{\mathbf{s}} : \rho \dashv\!\shortmid^{\mathsf{d}} \sigma$.*
Moreover if f is ρ-σ strict then $f = f^{\mathbf{s}}$, hence it is unique.

Proof. We design an algorithm $\mathbf{A}(f, \rho, \sigma, B)$ that takes as input an orchestrator f, the contracts ρ, σ, and a finite set of triples $B = \{(f_1, \rho_1, \sigma_1), \ldots, (f_n, \rho_n, \sigma_n)\}$, returning an orchestrator expression, as follows:

[3] We thank an anonymous referee for this remark.

[4] The definition of strict orchestrator is closely related to that of *relevant* orchestrator of Def. 3.3 in [11], where however the subcontract relation for the general contract formalism is investigated.

$$\begin{aligned}
\mathbf{A}(1,\rho,\sigma,B) &= 1\\
\mathbf{A}(\textstyle\bigvee_{i\in I}\mu_i.f_i,\rho,\sigma,B) &= \textstyle\bigvee\{\mu_k.\mathbf{A}(f_k,\rho_k,\sigma_k,B) \mid \rho \parallel_f \sigma \stackrel{\mu_k}{\Longrightarrow} \rho_k \parallel_{f_k} \sigma_k\}\\
\mathbf{A}(\mathsf{rec}\,x.f,\rho,\sigma,B) &= \begin{cases} x & \text{if } (\mathsf{rec}\,x.f,\rho,\sigma)\in B\\ \mathsf{rec}\,x.\mathbf{A}(f\{\mathsf{rec}\,x.f/x\},\rho,\sigma,B') & \text{otherwise}\end{cases}
\end{aligned}$$

where $B' = B \cup \{(\mathsf{rec}\,x.f,\rho,\sigma)\}$ and we assume that x is new (taking the alphabetic change $\mathsf{rec}\,y.f\{y/x\}$ for a new y otherwise).

Notice that in the above definition there is not necessarily any relation between the f_i in B and the f_i in $\bigvee_{i\in I}\mu_i.f_i$ (as a matter of fact, the former index ranges over $\{1,\ldots,n\}$ whereas the latter ranges over I, and the two sets of indexes do not necessarily coincide).

We recall that here we are identifying recursive orchestrators with their expansions. Since the LTS is finitely branching and recursion in both contracts and orchestrators is guarded, the set of triples $\{(\mu_k.f_k,\rho_k,\sigma_k) \mid \rho \parallel_f \sigma \stackrel{\mu_k}{\Longrightarrow} \rho_k \parallel_{f_k} \sigma_k\}$ is finite and computable. Any session contract or orchestrator can be seen as a regular tree. In what follows we hence identify the notion of subterm with that of subtree.

In any run of $\mathbf{A}(f,\rho,\sigma,B)$, all the subsequent calls $\mathbf{A}(f',\rho',\sigma',B')$ are with f', ρ' and σ' that are sub-terms of f, ρ and σ respectively, and the triples in B' either were in B, or are made of sub-terms of f, ρ and σ, so that the call $\mathbf{A}(f',\rho',\sigma',B'\cup\{(f',\rho',\sigma')\})$ is eventually met. Hence the second clause in the definition of $\mathbf{A}(\mathsf{rec}\,x.f,\rho,\sigma,B)$ cannot apply infinitely many times and the function $\mathbf{A}$ is well defined and total. Since $\bigvee\emptyset = 1$, $\mathbf{A}(f,\rho,\sigma,B)$ is always defined and yields an orchestrator.

By construction we have that if $\rho \parallel_f \sigma \stackrel{\mu}{\Longrightarrow} \rho' \parallel_{f'} \sigma'$ then for any B there exists $B' \supseteq B$ such that

$$\rho \parallel_{\mathbf{A}(f,\rho,\sigma,B)} \sigma \stackrel{\mu}{\Longrightarrow} \rho' \parallel_{\mathbf{A}(f',\rho',\sigma',B')} \sigma'$$

and vice versa. Whence the same statement for any finite $\boldsymbol{\mu}$, by induction over its length.

Take $f^{\mathbf{s}} = \mathbf{A}(f,\rho,\sigma,\emptyset)$. By construction $f^{\mathbf{s}} \stackrel{\boldsymbol{\mu}}{\longmapsto}$ if and only if $\rho \parallel_{f^{\mathbf{s}}} \sigma \stackrel{\boldsymbol{\mu}}{\Longrightarrow}$, i.e. $f^{\mathbf{s}}$ is ρ-σ-strict. By the above we conclude that if $f : \rho \dashv\!\mid^{\mathrm{d}} \sigma$ then $f^{\mathbf{s}} : \rho \dashv\!\mid^{\mathrm{d}} \sigma$.

Finally, if f is ρ-σ-strict then $\mathbf{A}(f,\rho,\sigma,B) = f$ for all B, hence in this case $f^{\mathbf{s}} = f$. □

We will now see how the procedure works on Example 2.14.

Example 2.16. Take

$$f = \mathsf{rec}\,x.(\langle\overline{a},a\rangle.x \vee \langle\varepsilon,c\rangle.x),\ \ \rho = \mathsf{rec}\,x.a.x,\ \text{ and } \sigma = \mathsf{rec}\,x.\overline{a}.\overline{c}.x;$$

then we have

$$\mathbf{A}(f,\rho,\sigma,\emptyset) = \mathsf{rec}\,x.\mathbf{A}(\langle\overline{a},a\rangle.f \vee \langle\varepsilon,c\rangle.f,\rho,\sigma,\{(f,\rho,\sigma)\})$$

Since $\rho \parallel_{\langle\overline{a},a\rangle.f\vee\langle\varepsilon,c\rangle.f} \sigma \stackrel{\langle\overline{a},a\rangle}{\Longrightarrow} \rho \parallel_f \overline{c}.\sigma$ but $\rho \parallel_{\langle\overline{a},a\rangle.f\vee\langle\varepsilon,c\rangle.f} \sigma \stackrel{\langle\varepsilon,c\rangle}{\not\Longrightarrow}$, we have

$$\mathbf{A}(\langle\overline{a},a\rangle.f \vee \langle\varepsilon,c\rangle.f,\rho,\sigma,\{(f,\rho,\sigma)\}) = \langle\overline{a},a\rangle.\mathbf{A}(f,\rho,\overline{c}.\sigma,\{(f,\rho,\sigma)\})$$

Observing that $(f,\rho,\overline{c}.\sigma) \neq (f,\rho,\sigma)$, we have

$$\mathbf{A}(f,\rho,\overline{c}.\sigma,\{(f,\rho,\sigma)\}) = \mathsf{rec}\,y.\mathbf{A}(\langle\overline{a},a\rangle.f \vee \langle\varepsilon,c\rangle.f,\rho,\overline{c}.\sigma,\{(f,\rho,\sigma),(f',\rho,\overline{c}.\sigma)\})$$

where $f' = \mathsf{rec}\,y.(\langle\overline{a},a\rangle.y \vee \langle\varepsilon,c\rangle.y)$ and y is a fresh variable.

This time $\rho \parallel_{\langle\overline{a},a\rangle.f\vee\langle\varepsilon,c\rangle.f} \overline{c}.\sigma \stackrel{\langle\varepsilon,c\rangle}{\Longrightarrow} \rho \parallel_f \sigma$ is the only possibility, so:

$$\mathbf{A}(\langle\overline{a},a\rangle.f \vee \langle\varepsilon,c\rangle.f,\rho,\overline{c}.\sigma,B) = \langle\varepsilon,c\rangle.\mathbf{A}(f,\rho,\sigma,B)$$

where $B = \{(f,\rho,\sigma),(f',\rho,\overline{c}.\sigma)\}$. Now $(f,\rho,\sigma) \in B$ and $f = \mathsf{rec}\,x.\cdots$ imply

$$\mathbf{A}(f,\rho,\sigma,B) = x$$

Putting all together we have

$$\mathbf{A}(f,\rho,\sigma,\emptyset) = \mathsf{rec}\,x.\langle\overline{a},a\rangle.\mathsf{rec}\,y.\langle\varepsilon,c\rangle.x$$

which is equivalent to $\mathsf{rec}\,x.\langle\overline{a},a\rangle.\langle\varepsilon,c\rangle.x$.

From now on, if not specified otherwise, we shall consider orchestrators to be strict.

$$(\text{Ax}): \frac{}{\Gamma \rhd \mathbf{1} : \mathbf{1} \dashv\!\dashv^{\mathsf{d}} \sigma} \qquad (\text{Hyp}): \frac{}{\Gamma, f : \rho \dashv\!\dashv^{\mathsf{d}} \sigma \rhd f : \rho \dashv\!\dashv^{\mathsf{d}} \sigma}$$

$$(\text{CpL}\Sigma\text{-L}): \frac{\Gamma' \rhd f' : \rho_p \dashv\!\dashv^{\mathsf{d}} \sigma}{\Gamma \rhd \langle \overline{a}_p, \varepsilon \rangle . f' : \Sigma_{i\in I} a_i . \rho_i \dashv\!\dashv^{\mathsf{d}} \sigma} (p \in I)$$

where $\Gamma' = \Gamma,\ \langle \overline{a}_p, \varepsilon \rangle . f' : \Sigma_{i\in I} a_i . \rho_i \dashv\!\dashv^{\mathsf{d}} \sigma$.

$$(\text{CpL}\Sigma\text{-R}): \frac{\Gamma' \rhd f' : \rho \dashv\!\dashv^{\mathsf{d}} \sigma_p}{\Gamma \rhd \langle \varepsilon, \overline{a}_p \rangle . f' : \rho \dashv\!\dashv^{\mathsf{d}} \Sigma_{i\in I} a_i . \sigma_i} (p \in I)$$

where $\Gamma' = \Gamma,\ \langle \varepsilon, \overline{a}_p \rangle . f' : \rho \dashv\!\dashv^{\mathsf{d}} \Sigma_{i\in I} a_i . \sigma_i$

$$(\text{CpL}\oplus\text{-R}): \frac{\forall j \in J.\, \Gamma' \rhd f_j : \rho \dashv\!\dashv^{\mathsf{d}} \sigma_j}{\Gamma \rhd \bigvee_{j\in J} \langle \varepsilon, b_j \rangle . f_j : \rho \dashv\!\dashv^{\mathsf{d}} \oplus_{j\in J} \overline{b}_j . \sigma_i}$$

where $\Gamma' = \Gamma,\ \bigvee_{j\in J} \langle \varepsilon, b_j \rangle . f_j : \rho \dashv\!\dashv^{\mathsf{d}} \oplus_{j\in J} \overline{b}_j . \sigma_j$

$$(\text{CpL}\oplus\text{-L}): \frac{\forall i \in I.\, \Gamma' \rhd f_i : \rho_i \dashv\!\dashv^{\mathsf{d}} \sigma}{\Gamma \rhd \bigvee_{i\in I} \langle a_i, \varepsilon \rangle . f_i : \oplus_{i\in I} \overline{a}_i . \rho_i \dashv\!\dashv^{\mathsf{d}} \sigma}$$

where $\Gamma' = \Gamma,\ \bigvee_{i\in I} \langle a_i, \varepsilon \rangle . f_i : \oplus_{i\in I} \overline{a}_i . \rho_i \dashv\!\dashv^{\mathsf{d}} \sigma$

$$(\text{CpL}\oplus\text{-}\Sigma): \frac{\Gamma' \rhd f_i : \rho_i \dashv\!\dashv^{\mathsf{d}} \Sigma_{j\in J} a_j . \sigma_j\ (\forall i \in H) \quad \Gamma' \rhd f_i : \rho_i \dashv\!\dashv^{\mathsf{d}} \sigma_i\ (\forall i \in K)}{\Gamma \rhd f : \oplus_{i\in H\cup K} \overline{a}_i . \rho_i \dashv\!\dashv^{\mathsf{d}} \Sigma_{j\in J} a_j . \sigma_j} (K \subseteq J)$$

where $f = \bigvee_{h\in H} \langle a_h, \varepsilon \rangle . f_h) \vee (\bigvee_{k\in K} \langle a_k, \overline{a}_k \rangle . f_k$
and $\Gamma' = \Gamma,\ f : \oplus_{i\in H\cup K} \overline{a}_i . \rho_i \dashv\!\dashv^{\mathsf{d}} \Sigma_{j\in J} a_j . \sigma_j$.

$$(\text{CpL}\Sigma\text{-}\oplus): \frac{\Gamma' \rhd f_j : \Sigma_{i\in I} a_i . \rho_i \dashv\!\dashv^{\mathsf{d}} \sigma_j\ (\forall j \in H) \quad \Gamma' \rhd f_j : \rho_j \dashv\!\dashv^{\mathsf{d}} \sigma_j\ (\forall j \in K)}{\Gamma \rhd f : \Sigma_{i\in I} a_i . \rho_i \dashv\!\dashv^{\mathsf{d}} \oplus_{j\in H\cup K} \overline{a}_j . \sigma_j} (K \subseteq I)$$

where $f = (\bigvee_{h\in H} \langle \varepsilon, a_h \rangle . f_h) \vee (\bigvee_{k\in K} \langle \overline{a}_k, a_k \rangle . f_k)$
and $\Gamma' = \Gamma,\ f : \Sigma_{i\in I} \overline{a}_i . \rho_i \dashv\!\dashv^{\mathsf{d}} \oplus_{j\in H\cup K} a_j . \sigma_j$.

Fig. 2. The formal system $\rhd$.

3. A formal system for $f : \rho \dashv\!\dashv^{\mathsf{d}} \sigma$ and its related decision procedure

As previously mentioned, the relation $f : \rho \dashv\!\dashv^{\mathsf{d}} \sigma$ is coarser than the proper orchestrated compliance relation $f : \rho \dashv\!\dashv \sigma$ that we shall define in the next section (Definition 4.6). Decidability of $f : \rho \dashv\!\dashv \sigma$ will strongly rely on decidability of $f : \rho \dashv\!\dashv^{\mathsf{d}} \sigma$, since $f : \rho \dashv\!\dashv \sigma$ holds whenever we have $f : \rho \dashv\!\dashv^{\mathsf{d}} \sigma$ and f is a *respectful* orchestrator. In this section we hence proceed to show that $f : \rho \dashv\!\dashv^{\mathsf{d}} \sigma$ is decidable, whereas in the next section, after formally defining the property of respectfulness, we shall prove it to be a decidable property.

We shall use 'coinductively compliant' as short for 'coinductively disrespectful orchestrated compliant'. We recall again that, when not specified otherwise, we take the equi-recursive view of recursion, by equating $\mathsf{rec}\, x.\sigma$ with $\sigma\{\mathsf{rec}\, x.\sigma / x\}$.

We will now define a formal system that we shall prove to axiomatically characterise the orchestrated compliance relation $f : \rho \dashv\!\dashv^{\mathsf{d}} \sigma$. The system is inspired to the coinductive axiomatisation of subtyping of the arrow and recursive-types in [15].

In the system below, the symbol $\dashv^{\mathsf{d}}$ will be used as syntactic counterpart of the relation $\dashv\!\dashv^{\mathsf{d}}$.

Definition 3.1 *(The formal system $\rhd$ for $\dashv\!\dashv^{\mathsf{d}}$).* We define a formal system $\rhd$ to derive judgements of the form $\Gamma \rhd f : \rho \dashv^{\mathsf{d}} \sigma$, where $\rho, \sigma \in \mathsf{SC}$, $f \in \mathsf{Orch}$ and Γ is a set of assumptions of the form $f_i : \rho_i \dashv^{\mathsf{d}} \sigma_i$.

Axioms and rules of the system are in Fig. 2; we write $\Gamma \rhd f : \rho \dashv^{\mathsf{d}} \sigma$ if this judgement is derivable using those and $\mathcal{D} :: \Gamma \rhd f : \rho \dashv^{\mathsf{d}} \sigma$ if we want to identify the derivation that shows it.

The system $\rhd$ can be shown to be decidable as a byproduct of the completeness property with respect to a suitable notion of semantics in which $\dashv^{\mathsf{d}}$ is interpreted as $\dashv\!\dashv^{\mathsf{d}}$. Hence it will follow that the relation $f : \rho \dashv\!\dashv^{\mathsf{d}} \sigma$ is decidable. Complete technical proofs and details concerning some results in the present section will be provided in Appendix A.

Remark 3.2. The system of Definition 3.1 is algorithmic (i.e. a decision procedure can be gotten out of its rules). In particular, the upward reconstruction of the derivation is syntax driven. Moreover, since the system satisfies a sort of subformula

property (see the proof of Lemma A.10), the derivation reconstruction can be proved to be terminating. The algorithm corresponding to the rules of the system is also deterministic, but for when it is possible to choose between the axiom (Hyp) and a rule. In such a case axiom (Hyp) has to take precedence in order to get termination.

The algorithmic nature of the formal system above will be implied by the soundness and completeness theorems (A.6 and 3.7) with respect to semantics provided in the following definition. These results can be proved along the lines of a similar proof in [7], which in turn extended the proofs for algorithmic subtyping in [14].

We can show that the formal system is sound with respect to the above mentioned semantics. As a matter of fact, in Appendix A a stronger version of the following theorem will be proved.

Theorem 3.3 *(Soundness). If* $\emptyset \rhd f : \rho \dashv^{\mathsf{d}} \sigma$ *then* $f : \rho \dashv\!\vdash^{\mathsf{d}} \sigma$.

Proof. See Proof of Theorem A.6 in Appendix A. □

We will now establish the completeness of the axiomatic system and decidability of derivability (and therefore of compliance) by means of the proof reconstruction algorithm **Prove** of Fig. 3.

Given a judgement $\Gamma \rhd f : \rho \dashv^{\mathsf{d}} \sigma$, if the algorithm **Prove** terminates, then it either returns a derivation $\mathcal{D}$ with conclusion $\Gamma \rhd f : \rho \dashv^{\mathsf{d}} \sigma$, or it returns **fail**. We will prove in fact in Lemma 3.6 that **Prove** always terminates.

The **Prove** algorithm tries to construct a proof for a given judgement by recursively proceeding bottom-up, each time applying the only possible rule that has the given judgement as conclusion, once it has been checked that rule (Hyp) does not apply. The algorithm fails as soon as the current judgement cannot be the conclusion of any rule.

It is not difficult to check that the algorithm **Prove** builds a derivation every time it does not fail.

Lemma 3.4. *If* **Prove**$(\Gamma \rhd f : \rho \dashv^{\mathsf{d}} \sigma) \neq$ **fail** *and* **Prove**$(\Gamma \rhd f : \rho \dashv^{\mathsf{d}} \sigma) = \mathcal{D}$, *then* $\mathcal{D} :: \Gamma \rhd f : \rho \dashv^{\mathsf{d}} \sigma$.

Proof. By construction and by induction over the tree of the recursive calls of **Prove**, which is finite when the execution terminates. □

The following lemma assures that a failure of the algorithm **Prove** can only happen if the configurations are not compliant.

Lemma 3.5. *If* **Prove**$(\Gamma \rhd f : \rho \dashv^{\mathsf{d}} \sigma) =$ **fail**, *then* $f : \rho \not\dashv\!\vdash^{\mathsf{d}} \sigma$.

Proof. See Appendix A. □

It is possible to show that **Prove** always terminates; the proof for this property will be provided in Appendix A and is inspired by the proof for decidability of the subtyping relation on recursive types in the π-calculus [16].

Lemma 3.6. *For all judgements* $\Gamma \rhd f : \rho \dashv^{\mathsf{d}} \sigma$ *the execution of* **Prove**$(\Gamma \rhd f : \rho \dashv^{\mathsf{d}} \sigma)$ *terminates.*

Proof. See Appendix A. □

Theorem 3.7 *(Completeness of* **Prove***). If* $f : \rho \dashv\!\vdash^{\mathsf{d}} \sigma$, *then* $\rhd f : \rho \dashv^{\mathsf{d}} \sigma$ *is derivable.*

Proof. By Lemma 3.5, $f : \rho \dashv\!\vdash^{\mathsf{d}} \sigma$ implies that **Prove**$(\rhd f : \rho \dashv^{\mathsf{d}} \sigma) \neq$ **fail**. Since by Lemma 3.6 the execution of **Prove**$(\rhd f : \rho \dashv^{\mathsf{d}} \sigma)$ terminates, we conclude by Lemma 3.4 that it produces a derivation $\mathcal{D}$ with conclusion $\rhd f : \rho \dashv^{\mathsf{d}} \sigma$. □

Corollary 3.8 *(Decidability of* $\dashv\!\vdash^{\mathsf{d}}$*). Given* f, ρ, *and* σ, $f : \rho \dashv\!\vdash^{\mathsf{d}} \sigma$ *is decidable.*

Proof. By Theorems A.6 and 3.7 $f : \rho \dashv\!\vdash^{\mathsf{d}} \sigma$ is equivalent to the derivability of $\rhd f : \rho \dashv^{\mathsf{d}} \sigma$, which is decidable by means of **Prove**. □

The following fact, which immediately follows by the first case of the **Prove** algorithm, will be useful when we will deal with orchestrator synthesis.

Fact 3.9. *Let* $\rhd f : \rho \dashv^{\mathsf{d}} \sigma$ *be a judgement and let* $\mathcal{D} =$ **Prove**$(\rhd f : \rho \dashv^{\mathsf{d}} \sigma) \neq$ **fail**.

For any judgement $\Gamma \rhd f' : \rho' \dashv^{\mathsf{d}} \sigma'$ *in* $\mathcal{D}$, *we have that* $\rho' \dashv^{\mathsf{d}} \sigma' \in \Gamma$ *if and only if* $\Gamma \rhd f' : \rho' \dashv^{\mathsf{d}} \sigma'$ *is an occurrence of the axiom* (Hyp) *in* $\mathcal{D}$.

Prove$(\Gamma \rhd f : \rho \dashv^{\mathsf{d}} \sigma)$ =

if $f : \rho \dashv^{\mathsf{d}} \sigma \in \Gamma$ **then** $\dfrac{}{\Gamma, f : \rho \dashv^{\mathsf{d}} \sigma \rhd f : \rho \dashv^{\mathsf{d}} \sigma}$ (Hyp)

else if $\rho = \mathbf{1}$ **and** $f = 1$ **then** $\dfrac{}{\Gamma \rhd 1 : \mathbf{1} \dashv^{\mathsf{d}} \sigma}$ (Ax)

else if $f = \langle \overline{a}_p, \varepsilon \rangle.f'$ **and** $\rho = \Sigma_{i\in I} a_i.\rho_i$ **and** $p \in I$ **then**

let $\Gamma' = \Gamma,\ \langle \overline{a}_p, \varepsilon \rangle.f' : \Sigma_{i\in I} a_i.\rho_i \dashv^{\mathsf{d}} \sigma$ **in**

let $\mathcal{D} =$ **Prove** $(\Gamma' \rhd f' : \rho_p \dashv^{\mathsf{d}} \sigma) \neq$ **fail in**

$$\dfrac{\boxed{\mathcal{D}}}{\Gamma \rhd \langle \overline{a}_p, \varepsilon \rangle.f' : \Sigma_{i\in I} a_i.\rho_i \dashv^{\mathsf{d}} \sigma}\ (\text{CPL}\Sigma\text{-L})$$

else if $f = \langle \varepsilon, \overline{a}_p \rangle.f'$ **and** $\sigma = \Sigma_{i\in I} a_i.\sigma_i$ **and** $p \in I$ **then**

let $\Gamma' = \Gamma,\ \langle \varepsilon, \overline{a}_p \rangle.f' : \rho \dashv^{\mathsf{d}} \Sigma_{i\in I} a_i.\sigma_i$ **in**

let $\mathcal{D} =$ **Prove** $(\Gamma' \rhd f' : \rho \dashv^{\mathsf{d}} \sigma_p) \neq$ **fail in**

$$\dfrac{\boxed{\mathcal{D}}}{\Gamma \rhd \langle \varepsilon, \overline{a}_p \rangle.f' : \rho \dashv^{\mathsf{d}} \Sigma_{i\in I} a_i.\sigma_i}\ (\text{CPL}\Sigma\text{-R})$$

else if $f = \bigvee_{i\in I} \langle a_i, \varepsilon \rangle.f_i$ **and** $\rho = \oplus_{i\in I} \overline{a}_i.\rho_i$ **then**

let $\Gamma' = \Gamma,\ \bigvee_{i\in I} \langle a_i, \varepsilon \rangle.f_i : \oplus_{i\in I} \overline{a}_i.\rho_i$ **in**

foreach $i \in I$ **let** $\mathcal{D}_i =$ **Prove** $(\Gamma' \rhd f_i : \rho_i \dashv^{\mathsf{d}} \sigma) \neq$ **fail in**

$$\dfrac{\boxed{\mathcal{D}_i \quad (\forall i \in I)}}{\Gamma \rhd \bigvee_{i\in I} \langle a_i, \varepsilon \rangle.f_i : \oplus_{i\in I} \overline{a}_i.\rho_i \dashv^{\mathsf{d}} \sigma}\ (\text{CPL}\oplus\text{-L})$$

else if $f = \bigvee_{j\in J} \langle \varepsilon, b_j \rangle.f_j$ **and** $\sigma = \oplus_{j\in J} \overline{b}_j.\sigma_i$ **then**

let $\Gamma' = \Gamma,\ \bigvee_{j\in J} \langle \varepsilon, b_j \rangle.f_j : \rho \dashv^{\mathsf{d}} \oplus_{j\in J} \overline{b}_j.\sigma_i$ **in**

foreach $j \in J$ **let** $\mathcal{D}_j =$ **Prove** $(\Gamma' \rhd f_j : \rho \dashv^{\mathsf{d}} \sigma_j) \neq$ **fail in**

$$\dfrac{\boxed{\mathcal{D}_j \quad (\forall j \in J)}}{\Gamma \rhd \bigvee_{j\in J} \langle \varepsilon, b_j \rangle.f_j : \rho \dashv^{\mathsf{d}} \oplus_{j\in J} \overline{b}_j.\sigma_i}\ (\text{CPL}\oplus\text{-R})$$

else if $f = \bigvee_{h\in H} \langle a_h, \varepsilon \rangle.f_h) \vee (\bigvee_{k\in K} \langle a_k, \overline{a}_k \rangle.f_k$

and $\rho = \oplus_{i\in H\cup K} \overline{a}_i.\rho_i$ **and** $\sigma = \Sigma_{j\in J} a_j.\sigma_j$ **and** $K \subseteq J$ **then**

let $\Gamma' = \Gamma,\ \Gamma \rhd \bigvee_{h\in H} \langle a_h, \varepsilon \rangle.f_h) \vee (\bigvee_{k\in K} \langle a_k, \overline{a}_k \rangle.f_k : \oplus_{i\in H\cup K} \overline{a}_i.\rho_i \dashv^{\mathsf{d}} \Sigma_{j\in J} a_j.\sigma_j$ **in**

foreach $i \in H$ **let** $\mathcal{D}_i =$ **Prove** $(\Gamma' \rhd f_i : \rho_i \dashv^{\mathsf{d}} \Sigma_{j\in J} a_j.\sigma_j) \neq$ **fail in**

foreach $i \in K$ **let** $\mathcal{D}_i =$ **Prove** $(\Gamma' \rhd f_i : \rho_i \dashv^{\mathsf{d}} \sigma_i) \neq$ **fail in**

$$\dfrac{\boxed{\mathcal{D}_i \quad (\forall i \in H)} \qquad \boxed{\mathcal{D}_i \quad (\forall i \in K)}}{\Gamma \rhd \bigvee_{h\in H} \langle a_h, \varepsilon \rangle.f_h) \vee (\bigvee_{k\in K} \langle a_k, \overline{a}_k \rangle.f_k : \oplus_{i\in H\cup K} \overline{a}_i.\rho_i \dashv^{\mathsf{d}} \Sigma_{j\in J} a_j.\sigma_j}\ (\text{CPL}\oplus\text{-}\Sigma)$$

else if $f = \bigvee_{h\in H} \langle \varepsilon, a_h \rangle.f_h) \vee (\bigvee_{k\in K} \langle \overline{a}_k, a_k \rangle.f_k$

and $\rho = \Sigma_{i\in I} a_i.\rho_i$ **and** $\sigma = \oplus_{j\in H\cup K} \overline{a}_j.\sigma_j$ **and** $K \subseteq I$ **then**

let $\Gamma' = \Gamma,\ \bigvee_{h\in H} \langle \varepsilon, a_h \rangle.f_h) \vee (\bigvee_{k\in K} \langle \overline{a}_k, a_k \rangle.f_k : \Sigma_{i\in I} a_i.\rho_i \dashv^{\mathsf{d}} \oplus_{j\in H\cup K} \overline{a}_j.\sigma_j$ **in**

foreach $j \in H$ **let** $\mathcal{D}_j =$ **Prove** $(\Gamma' \rhd f_j : \Sigma_{i\in I} a_i.\rho_i \dashv^{\mathsf{d}} \sigma_j) \neq$ **fail in**

foreach $j \in K$ **let** $\mathcal{D}_j =$ **Prove** $(\Gamma' \rhd f_j : \rho_j \dashv^{\mathsf{d}} \sigma_j) \neq$ **fail in**

$$\dfrac{\boxed{\mathcal{D}_j \quad (\forall j \in H)} \qquad \boxed{\mathcal{D}_j \quad (\forall j \in K)}}{\Gamma \rhd \bigvee_{h\in H} \langle \varepsilon, a_h \rangle.f_h) \vee (\bigvee_{k\in K} \langle \overline{a}_k, a_k \rangle.f_k : \Sigma_{i\in I} a_i.\rho_i \dashv^{\mathsf{d}} \oplus_{j\in H\cup K} \overline{a}_j.\sigma_j}\ (\text{CPL}\Sigma\text{-}\oplus)$$

else fail

Fig. 3. The algorithm **Prove**.

Let us look at the result of the **Prove** algorithm on the 'meteorological' example of the Introduction.

Example 3.10. Let

$$\begin{aligned}
\rho &= \mathsf{rec}\,x.\overline{\mathtt{tR}}.\overline{\mathtt{hR}}.\mathtt{t}.\mathtt{h}.x \\
\sigma &= \mathsf{rec}\,x.\mathtt{tR}.\mathtt{hR}.(\overline{\mathtt{t}}.\overline{\mathtt{h}}.\overline{\mathtt{w}}.x \oplus \overline{\mathtt{h}}.\overline{\mathtt{t}}.\overline{\mathtt{w}}.x), \text{ and} \\
\mathsf{f} &= \mathsf{rec}\,x.\langle \mathtt{tR}, \overline{\mathtt{tR}} \rangle.\langle \mathtt{hR}, \overline{\mathtt{hR}} \rangle.(\langle \overline{\mathtt{t}}, \mathtt{t} \rangle.\langle \overline{\mathtt{h}}, \mathtt{h} \rangle.\langle \varepsilon, \mathtt{w} \rangle.\mathtt{x} \vee \langle \varepsilon, \mathtt{h} \rangle.\langle \overline{\mathtt{t}}, \mathtt{t} \rangle.\langle \overline{\mathtt{h}}, \varepsilon \rangle\langle \varepsilon, \mathtt{w} \rangle.\mathtt{x})
\end{aligned}$$

Then

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\gamma_0,\gamma_1,\gamma_2,\gamma_3,\gamma_4 \rhd \mathsf{f}:\rho \dashv^{\mathsf{d}} \sigma}\,(\textsc{Hyp})}{\gamma_0,\gamma_1,\gamma_2,\gamma_3 \rhd \mathsf{f}_4:\rho \dashv^{\mathsf{d}} \overline{\mathtt{w}}.\sigma}\,(\textsc{Cpl}\oplus\text{-R})}{\gamma_0,\gamma_1,\gamma_2 \rhd \mathsf{f}_3:\mathtt{h}.\rho \dashv^{\mathsf{d}} \overline{\mathtt{h}}.\overline{\mathtt{w}}.\sigma}\,(\textsc{Cpl}\Sigma\text{-}\oplus) \quad \dfrac{\dfrac{\dfrac{}{\gamma_0,\gamma_1,\gamma_2,\gamma_5,\gamma_6,\gamma_7 \rhd \mathsf{f}:\rho \dashv^{\mathsf{d}} \sigma}\,(\textsc{Hyp})}{\gamma_0,\gamma_1,\gamma_2,\gamma_5,\gamma_6 \rhd \mathsf{f}_7:\rho \dashv^{\mathsf{d}} \overline{\mathtt{w}}.\sigma}\,(\textsc{Cpl}\oplus\text{-R})}{\gamma_0,\gamma_1,\gamma_2,\gamma_5 \rhd \mathsf{f}_6:\mathtt{h}.\rho \dashv^{\mathsf{d}} \overline{\mathtt{w}}.\sigma}\,(\textsc{Cpl}\Sigma\text{-L})}{\gamma_0,\gamma_1,\gamma_2 \rhd \mathsf{f}_5:\mathtt{t}.\mathtt{h}.\rho \dashv^{\mathsf{d}} \overline{\mathtt{t}}.\overline{\mathtt{w}}.\sigma}\,(\textsc{Cpl}\Sigma\text{-}\oplus)}{\gamma_0,\gamma_1 \rhd \mathsf{f}_2:\mathtt{t}.\mathtt{h}.\rho \dashv^{\mathsf{d}} (\overline{\mathtt{t}}.\overline{\mathtt{h}}.\overline{\mathtt{w}}.\sigma \oplus \overline{\mathtt{h}}.\overline{\mathtt{t}}.\overline{\mathtt{w}}.\sigma)}\,(\textsc{Cpl}\oplus\text{-R})}{\gamma_0 \rhd \mathsf{f}_1:\overline{\mathtt{hR}}.\mathtt{t}.\mathtt{h}.\rho \dashv^{\mathsf{d}} \mathtt{hR}.(\overline{\mathtt{t}}.\overline{\mathtt{h}}.\overline{\mathtt{w}}.\sigma \oplus \overline{\mathtt{h}}.\overline{\mathtt{t}}.\overline{\mathtt{w}}.\sigma)}\,(\textsc{Cpl}\oplus\text{-R})}{\gamma_0 \rhd \mathsf{f}_1:\rho \dashv^{\mathsf{d}} \sigma}\,(\textsc{Cpl}\oplus\text{-R})$$

where

$$\begin{array}{ll}
\mathsf{f}_1 = \langle \mathtt{hR}, \overline{\mathtt{hR}}\rangle.(\langle \overline{\mathtt{t}}, \mathtt{t}\rangle.\langle \overline{\mathtt{h}}, \mathtt{h}\rangle.\langle \varepsilon, \mathtt{w}\rangle.\mathsf{f} & \gamma_0 = \mathsf{f}:\rho \dashv^{\mathsf{d}} \sigma \\
\quad \vee \langle \varepsilon, \mathtt{h}\rangle.\langle \overline{\mathtt{t}}, \mathtt{t}\rangle.\langle \overline{\mathtt{h}}, \varepsilon\rangle\langle \varepsilon, \mathtt{w}\rangle.\mathsf{f}) & \gamma_1 = \mathsf{f}_1:\overline{\mathtt{hR}}.\mathtt{t}.\mathtt{h}.\rho \dashv^{\mathsf{d}} \mathtt{hR}.(\overline{\mathtt{t}}.\overline{\mathtt{h}}.\overline{\mathtt{w}}.\sigma \oplus \overline{\mathtt{h}}.\overline{\mathtt{t}}.\overline{\mathtt{w}}.\sigma) \\
\mathsf{f}_2 = (\langle \overline{\mathtt{t}}, \mathtt{t}\rangle.\langle \overline{\mathtt{h}}, \mathtt{h}\rangle.\langle \varepsilon, \mathtt{w}\rangle.\mathsf{f} & \gamma_2 = \mathsf{f}_2:\mathtt{t}.\mathtt{h}.\rho \dashv^{\mathsf{d}} (\overline{\mathtt{t}}.\overline{\mathtt{h}}.\overline{\mathtt{w}}.\sigma \oplus \overline{\mathtt{h}}.\overline{\mathtt{t}}.\overline{\mathtt{w}}.\sigma) \\
\quad \vee \langle \varepsilon, \mathtt{h}\rangle.\langle \overline{\mathtt{t}}, \mathtt{t}\rangle.\langle \overline{\mathtt{h}}, \varepsilon\rangle\langle \varepsilon, \mathtt{w}\rangle.\mathsf{f}) & \gamma_3 = \mathsf{f}_3:\mathtt{h}.\rho \dashv^{\mathsf{d}} \overline{\mathtt{h}}.\overline{\mathtt{w}}.\sigma \\
\mathsf{f}_3 = \langle \overline{\mathtt{h}}, \mathtt{h}\rangle.\langle \varepsilon, \mathtt{w}\rangle.\mathsf{f} & \gamma_4 = \mathsf{f}_4:\rho \dashv^{\mathsf{d}} \overline{\mathtt{w}}.\sigma \\
\mathsf{f}_4 = \langle \varepsilon, \mathtt{w}\rangle.\mathsf{f} & \gamma_5 = \mathsf{f}_5:\mathtt{t}.\mathtt{h}.\rho \dashv^{\mathsf{d}} \overline{\mathtt{t}}.\overline{\mathtt{w}}.\sigma \\
\mathsf{f}_5 = \langle \overline{\mathtt{t}}, \mathtt{t}\rangle.\langle \overline{\mathtt{h}}, \varepsilon\rangle\langle \varepsilon, \mathtt{w}\rangle.\mathsf{f} & \gamma_6 = \mathsf{f}_6:\mathtt{h}.\rho \dashv^{\mathsf{d}} \overline{\mathtt{w}}.\sigma \\
\mathsf{f}_6 = \langle \overline{\mathtt{h}}, \varepsilon\rangle\langle \varepsilon, \mathtt{w}\rangle.\mathsf{f} & \gamma_7 = \mathsf{f}_7:\rho \dashv^{\mathsf{d}} \overline{\mathtt{w}}.\sigma \\
\mathsf{f}_7 = \langle \varepsilon, \mathtt{w}\rangle.\mathsf{f} &
\end{array}$$

The following is another non-trivial example of a derivation, which can be obtained by the algorithm **Prove**.

Example 3.11. Let us consider a client ρ, a server $\overline{c}.\sigma$ and an orchestrator f, where

$$\begin{array}{rcl}
\rho & = & \mathsf{rec}\, x.b.c.x, \\
\sigma & = & \mathsf{rec}\, y.(\overline{c}.\overline{b}.\overline{a}.y \oplus \overline{b}.\mathsf{rec}\, x.\overline{b}.\overline{c}.x), \text{ and} \\
f & = & \langle \varepsilon, c\rangle.\mathsf{rec}\, z.(\langle \varepsilon, c\rangle.\langle \overline{b}, b\rangle.\langle \overline{c}, \varepsilon\rangle.\langle \varepsilon, a\rangle.z \vee \langle \overline{b}, b\rangle.\langle \overline{c}, \varepsilon\rangle.\mathsf{rec}\, w.\langle \overline{b}, b\rangle.\langle \overline{c}, c\rangle.w).
\end{array}$$

Then we have

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\gamma_0,\gamma_1,\gamma_2,\gamma_3,\gamma_4 \rhd f_1:\rho \dashv^{\mathsf{d}} \sigma}\,(\textsc{Hyp})}{\gamma_0,\gamma_1,\gamma_2,\gamma_3 \rhd \langle \varepsilon, a\rangle.f_1:\rho \dashv^{\mathsf{d}} \overline{a}.\sigma}\,(\textsc{Cpl}\oplus\text{-R})}{\gamma_0,\gamma_1,\gamma_2 \rhd \langle \overline{c}, \varepsilon\rangle\langle \varepsilon, a\rangle.f_1:c.\rho \dashv^{\mathsf{d}} \overline{a}.\sigma}\,(\textsc{Cpl}\Sigma\text{-L})}{\gamma_0,\gamma_1 \rhd \langle \overline{b}, b\rangle\langle \overline{c}, \varepsilon\rangle\langle \varepsilon, a\rangle.f_1:b.c.\rho \dashv^{\mathsf{d}} \overline{b}.\overline{a}.\sigma}\,(\textsc{Cpl}\Sigma\text{-}\oplus)$$

and

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\gamma_0,\gamma_1,\gamma_5,\gamma_6,\gamma_7 \rhd f_2:\rho \dashv^{\mathsf{d}} \sigma_1}\,(\textsc{Hyp})}{\gamma_0,\gamma_1,\gamma_5,\gamma_6 \rhd \langle \overline{c}, c\rangle.f_2:c.\rho \dashv^{\mathsf{d}} \overline{c}.\sigma_1}\,(\textsc{Cpl}\Sigma\text{-}\oplus)}{\gamma_0,\gamma_1,\gamma_5 \rhd \langle \overline{b}, b\rangle.\langle \overline{c}, c\rangle.f_2:b.c.\rho \dashv^{\mathsf{d}} \overline{b}.\overline{c}.\sigma_1}\,(\textsc{Cpl}\Sigma\text{-}\oplus)}{\gamma_0,\gamma_1 \rhd \langle \overline{c}, \varepsilon\rangle.f_2:c.\rho \dashv^{\mathsf{d}} \mathsf{rec}\, x.\overline{b}.\overline{c}.x}\,(\textsc{Cpl}\Sigma\text{-L})$$

from which we conclude

$$\dfrac{\dfrac{\gamma_0,\gamma_1 \rhd \langle \overline{b}, b\rangle\langle \overline{c}, \varepsilon\rangle\langle \varepsilon, a\rangle.f_1:b.c.\rho \dashv^{\mathsf{d}} \overline{b}.\overline{a}.\sigma \quad \gamma_0,\gamma_1 \rhd \langle \overline{c}, \varepsilon\rangle.f_2:c.\rho \dashv^{\mathsf{d}} \mathsf{rec}\, x.\overline{b}.\overline{c}.x}{\gamma_0 \rhd \langle \varepsilon, c\rangle\langle \overline{b}, b\rangle\langle \overline{c}, \varepsilon\rangle\langle \varepsilon, a\rangle.f_1 \vee \langle \overline{b}, b\rangle.\langle \overline{c}, \varepsilon\rangle.f_2):b.c.\rho \dashv^{\mathsf{d}} \overline{c}.\overline{b}.\overline{a}.\sigma \oplus \overline{b}.\mathsf{rec}\, x.\overline{b}.\overline{c}.x}\,(\textsc{Cpl}\Sigma\text{-}\oplus)}{\rhd \langle \varepsilon, c\rangle.f_1:\mathsf{rec}\, x.b.c.x \dashv^{\mathsf{d}} \overline{c}.\mathsf{rec}\, y.(\overline{c}.\overline{b}.\overline{a}.y \oplus \overline{b}.\mathsf{rec}\, x.\overline{b}.\overline{c}.x)}\,(\textsc{Cpl}\oplus\text{-R})$$

where

$$\begin{array}{l}
\gamma_0 = f:\rho \dashv^{\mathsf{d}} \overline{c}.\sigma \\
\gamma_1 = \langle \varepsilon, c\rangle\langle \overline{b}, b\rangle\langle \overline{c}, \varepsilon\rangle\langle \varepsilon, a\rangle.f_1 \vee \langle \overline{b}, b\rangle.\langle \overline{c}, \varepsilon\rangle.f_2):b.c.\rho \dashv^{\mathsf{d}} \overline{c}.\overline{b}.\overline{a}.\sigma \oplus \overline{b}.\mathsf{rec}\, x.\overline{b}.\overline{c}.x \\
\gamma_4 = \langle \varepsilon, a\rangle.f_1:\rho \dashv^{\mathsf{d}} \overline{a}.\sigma \\
\gamma_2 = \langle \overline{b}, b\rangle\langle \overline{c}, \varepsilon\rangle\langle \varepsilon, a\rangle.f_1:b.c.\rho \dashv^{\mathsf{d}} \overline{b}.\overline{a}.\sigma \\
\gamma_5 = \langle \overline{c}, \varepsilon\rangle.f_2:c.\rho \dashv^{\mathsf{d}} \mathsf{rec}\, x.\overline{b}.\overline{c}.x \\
\gamma_3 = \langle \overline{c}, \varepsilon\rangle\langle \varepsilon, a\rangle.f_1:c.\rho \dashv^{\mathsf{d}} \overline{a}.\sigma \\
\gamma_6 = \langle \overline{b}, b\rangle.\langle \overline{c}, c\rangle.f_2:b.c.\rho \dashv^{\mathsf{d}} \overline{b}.\overline{c}.\sigma_1 \\
\gamma_7 = \langle \overline{c}, c\rangle.f_2:c.\rho \dashv^{\mathsf{d}} \overline{c}.\sigma_1
\end{array}$$

and

$$f_1 = \mathsf{rec}\, z.(\langle \varepsilon, c\rangle.\langle \overline{b}, b\rangle.\langle \overline{c}, \varepsilon\rangle.\langle \varepsilon, a\rangle.z \vee \langle \overline{b}, b\rangle.\langle \overline{c}, \varepsilon\rangle.\mathsf{rec}\, w.\langle \overline{b}, b\rangle.\langle \overline{c}, c\rangle.w)$$
$$f_2 = \mathsf{rec}\, w.\langle \overline{b}, b\rangle.\langle \overline{c}, c\rangle.w$$
$$\sigma_1 = \mathsf{rec}\, x.\overline{b}.\overline{c}.x$$

Remark 3.12 *(Complexity issues).* The algorithm **Prove** (and hence the decision procedure for compliance) is extremely simple, but its complexity is exponential. However, the complexity of the compliance decision procedure can be reduced down to a polynomial complexity by adapting the approach proposed in [17], in particular by adapting the concrete subtyping algorithm for recursive arrow and product types in §11 of [17].

The pruning relation $\leq$ on orchestrators Consider the orchestrator

$$f \;=\; \mathsf{rec}\, x.\langle a, \varepsilon\rangle.\mathbf{1} \vee \langle b, \overline{b}\rangle.x$$

which is such that

$$f : \mathsf{rec}\, x.\,\overline{a} \oplus \overline{b}.x \dashv\!\!\!\vdash^{\mathsf{d}} \mathsf{rec}\, x.\,b.x$$

Also

$$\mathsf{rec}\, x.\langle a, \varepsilon\rangle.\langle \varepsilon, \overline{d}\rangle.\mathbf{1} \vee \langle b, \overline{b}\rangle.x \vee \langle c, \varepsilon\rangle.x$$

is an orchestrator for the given client and server although it keeps on offering orchestration actions even when f would stop, and provides more choices than f. We formalise this relation between orchestrators via a preorder which we call the pruning relation.

Definition 3.13 *(Pruning relation $\leq$).* We define $\leq \subseteq \mathsf{Orch} \times \mathsf{Orch}$ by

$$f \leq f' \text{ if } \emptyset \rhd_{\leq} f \leq f'$$

is derivable through the inference system

$$(\mathrm{Ax}) : \frac{}{\Gamma \rhd_{\leq} \mathbf{1} \leq f} \qquad (\mathrm{Id}) : \frac{}{\Gamma \rhd_{\leq} f \leq f} \qquad (\mathrm{Hyp}) : \frac{}{\Gamma, f \leq f' \rhd_{\leq} f \leq f'}$$

$$(\mu) : \frac{\Gamma, o.f \leq o'.f' \rhd_{\leq} f \leq f'}{\Gamma \rhd_{\leq} o.f \leq o'.f'} \qquad (\vee) : \frac{\Gamma, \bigvee_{i\in I} f_i \leq \bigvee_{j\in J} f'_j \rhd_{\leq} f_i \leq f'_i \quad (\forall i \in I)}{\Gamma \rhd_{\leq} \bigvee_{i\in I} f_i \leq \bigvee_{j\in J} f'_j}\ (I \subseteq J)$$

For example, we have:

$$\mathsf{rec}\, x.\langle a, \varepsilon\rangle.\mathbf{1} \vee \langle b, \overline{b}\rangle.x \;\leq\; \mathsf{rec}\, x.\langle a, \varepsilon\rangle.\langle \varepsilon, \overline{d}\rangle.\mathbf{1} \vee \langle b, \overline{b}\rangle.x \vee \langle c, \varepsilon\rangle.x.$$

We conclude this section by showing a lemma relating pruning to disrespectful compliance.

Lemma 3.14. *Let $f_1 \leq f_2$ and let $\boldsymbol{\mu}$ be such that $f_1 \overset{\boldsymbol{\mu}}{\longmapsto} g_1$ and $f_2 \overset{\boldsymbol{\mu}}{\longmapsto} g_2$. Then only the following cases can occur:*

- *$g_1 = o.g'_1$ and $g_2 = o.g'_2$, for some o, g'_1, and g'_2.*
- *$g_1 = \bigvee_{i\in I} g'_i$ and $g_2 = \bigvee_{j\in J} g'_j$, with $I \subseteq J$.*

Proof. By definition of $\leq$, taking into account the condition of input-disjointness imposed on orchestrators in Definition 2.7. □

Lemma 3.15. *For any f, f', if $f \leq f'$ (possibly not $\rho \cdot \sigma$ strict), then $\mathsf{Tr}(f) \subseteq \mathsf{Tr}(f')$. Moreover, for all ρ, σ:*

$$f : \rho \dashv\!\!\!\vdash^{\mathsf{d}} \sigma \;\;\&\;\; f \leq f' \;\;\Longrightarrow\;\; f' : \rho \dashv\!\!\!\vdash^{\mathsf{d}} \sigma$$

Proof. If $f \leq f'$ then f' allows longer interactions than f by (Ax) and (μ), and offers more alternatives than f by $(\vee)$. Since the set of traces of orchestrators are prefix-closed it follows that $\mathsf{Tr}(f) \subseteq \mathsf{Tr}(f')$.

By contraposition, assume $f \leq f'$ and $f' : \rho \not\dashv\!\!\!\vdash^{\mathsf{d}} \sigma$. By definition, the latter means that $\rho \parallel_{f'} \sigma \overset{\boldsymbol{\mu}}{\Longrightarrow} \rho' \parallel_{g'} \sigma' \not\longrightarrow$ for a certain g', a $\boldsymbol{\mu} \in \mathsf{Tr}(f')$, and where $\rho' \neq \mathbf{1}$. In case $\boldsymbol{\mu} \in \mathsf{Tr}(f)$, it immediately follows that $f : \rho \not\dashv\!\!\!\vdash^{\mathsf{d}} \sigma$. Otherwise, let $\boldsymbol{\mu'}$ be the longest prefix of $\boldsymbol{\mu}$ such that $\boldsymbol{\mu'} \in \mathsf{Tr}(f)$; consider the longest reduction sequences such that $\rho \parallel_{f'} \sigma \overset{\boldsymbol{\mu'}}{\Longrightarrow} \rho'' \parallel_{g''} \sigma''$ and $\rho \parallel_{f} \sigma \overset{\boldsymbol{\mu'}}{\Longrightarrow} \rho'' \parallel_{\tilde{g}} \sigma''$. By Lemma 3.14, we have to consider just three possible cases:

- $g'' = o.g_1'$ and $\tilde{g} = o.g_2'$ for some o, g_1' and g_2'. This case cannot actually occur, otherwise $\boldsymbol{\mu'}$ would not be the longest prefix of $\boldsymbol{\mu}$ such that $\boldsymbol{\mu'} \in \mathrm{Tr}(f)$.
- $\tilde{g} = \bigvee_{i\in I}\langle a_i, \phi_i\rangle.g_i'$ and $g'' = \bigvee_{j\in J}\langle a_j, \phi_j\rangle.g_j'$ with $I \subseteq J$, and where $\rho'' = \overline{a_k}.\rho_k$ with $k \in J \setminus I$. Then we get $\rho \parallel_f \sigma \stackrel{\boldsymbol{\mu'}}{\Longrightarrow} \rho'' \parallel_{\tilde{g}} \sigma'' \not\rightarrow$, implying that $f : \rho \not\dashv^{\mathsf{d}} \sigma$.
- $\tilde{g} = \bigvee_{i\in I}\langle \phi_i, a_i\rangle.g_i'$ and $g'' = \bigvee_{j\in J}\langle \phi_j, a_j\rangle.g_j'$ with $I \subseteq J$, and where $\sigma'' = \overline{a_k}.\sigma_k$ with $k \in J \setminus I$. This case is similar to the previous one.

where $\phi_i, \phi_j \in \{\varepsilon\} \cup \overline{\mathcal{N}}$. □

As we mentioned before, the notion of orchestrated compliance cannot coincide with that of disrespectful orchestrated compliance. As a matter of fact, for any client/server pair some suitable orchestrator can be found which makes the client and server disrespectfully orchestrated compliant. We wish instead a server to be orchestrated compliant with a client only in case the orchestrator f mediating their interaction does not show any "unfair" behaviour. We call such orchestrators *respectful*. In the next section we will discuss what are the unfair behaviours we wish to prevent in orchestrators and will formally define the notion of respectfulness for orchestrators. From this we shall be able to provide the formal definition of orchestrated compliance. Its decidability will follow as a corollary of both the decidability of the disrespectful orchestrated compliance we have just proved in the present section, and the decidability of the respectfulness property, that will be also proved in the next section.

4. Respectfulness and orchestrated session compliance

In Definition 2.11 we considered the relation $\dashv^{\mathsf{d}}$, which we have studied so far. This is however much weaker than expected, since it admits orchestrators with "unfair" behaviour. We will now address this issue.

Consider the following simple orchestrated system

$$\overline{a}.\overline{b} \parallel_f a.c.d \text{ where } f = \langle a, \overline{a}\rangle.\langle b, \varepsilon\rangle.\mathbf{1}.$$

It is straightforward to check that $f : \overline{a}.\overline{b} \dashv^{\mathsf{d}} a.c.d$ since

$$\overline{a}.\overline{b} \parallel_f a.c.d \stackrel{\boldsymbol{\mu}}{\Longrightarrow} \mathbf{1} \parallel_1 c.d \not\rightarrow, \text{ with } \boldsymbol{\mu} = \langle a, \overline{a}\rangle\langle b, \varepsilon\rangle$$

is the only possible trace. It is certainly true that all the client's requests have been 'satisfied', but not all by the server. In particular, the action $\overline{b}$ represents an output by the client which never reaches the server, so that the orchestrator has not acted as a fair mediator; rather it has deceived the client by a false reaction.

We can prevent the above unfair behaviour by modifying Definition 2.11 (i) when $\rho' \parallel_{f'} \sigma' \not\rightarrow$. In particular, we will impose a condition assuring the client-to-server buffer associated to f' to be empty (as will be formalised below, orchestrators have buffering capabilities, a client-to-server one and a server-to-client one). Of course, a similar condition must be imposed also to take into account infinite interactions; in fact, in an infinite interaction, for any possible name a used by the orchestrator, the latter cannot indefinitely perform input actions for a from the client (even if interspersed with actions for other names) without ever delivering an a to the server. We must therefore forbid a client like $\mathsf{rec}\,x.\overline{a}.\overline{c}.x$ to be compliant with the server $\mathsf{rec}\,x.c.x$ by means of the orchestrator $\mathsf{rec}\,x.\langle a, \varepsilon\rangle.\langle c, \overline{c}\rangle.x$. Orchestrated finite and infinite interaction sequences which avoid unwanted situations like those just sketched will be called **client-respectful**.

Even if the notion of compliance enforces the sense of the bias towards the client (every client request must be eventually satisfied by the server), some conditions need to be imposed on the part of interactions on behalf of the server. In fact, we wish to prevent a server to be compliant with a client by means of an orchestrator that, from a certain moment on, interacts infinitely many times with the server only, like in the orchestrated system:

$$\overline{a}.b \parallel_f \mathsf{rec}\,x.\overline{c}.\overline{b}.x \text{ where } f = \langle a, \varepsilon\rangle.\mathsf{rec}\,x.\langle \varepsilon, c\rangle.\langle \varepsilon, b\rangle.x$$

We wish to prevent this kind of infinite interaction that we dub **definitely server-inputted**. Notice that, however, we do allow interactions in which the orchestrator can perform the input of some a from the server infinitely many times, without ever performing an output of a to the client, as happens for `wind` in the example in the introduction.

The above properties do share similarities with undecidable properties like, for example, termination of two-counter machines [18]. The restrictions imposed on interactions by our session-contracts setting, however, manage to maintain such properties into the realm of decidability, as we shall prove below. As far as our knowledge goes, it is not known whether decidability can be obtained also for the general notion of contract.

Among the properties we have to take care of, one is that in an interaction sequence there cannot exist an orchestrator action removing an element from an empty buffer, i.e. a **sound** sequence never sends an element a to a server or to a client if an a has not been previously received. For instance, the orchestrator $\langle \varepsilon, \overline{b}\rangle.\langle \varepsilon, \overline{a}\rangle.\langle a, \varepsilon\rangle.\langle b, \varepsilon\rangle$ is not sound since it sends two messages a and b to the server before having previously received them. On the contrary, the orchestrator $\langle a, \varepsilon\rangle.\langle b, \varepsilon\rangle.\langle \varepsilon, \overline{b}\rangle.\langle \varepsilon, \overline{a}\rangle$ is sound.

We call the combination of all the above properties **respectfulness**. Summarising:

$$\begin{aligned}\textbf{respectfulness} &= \textbf{soundness}\\ &+ \textbf{client-respectfulness}\\ &+ \textbf{non-definitely server-inputted-ness}\end{aligned}$$

where, intuitively,

soundness: only messages previously received can be sent;
client-respectfulness: any message received from the client is eventually delivered to the server;
non-definitely server-inputted-ness: no interaction can eventually consist just of getting messages from the server.

Notice that the unwanted sort of interaction consisting of indefinitely exchanging input and output messages with the server from a certain point on is implicitly prevented by the property of soundness.

As a first step toward the formal definition of respectfulness, we will define the notion of orchestrator buffer, since respectfulness will be based on it.

Buffers Orchestrators have buffering capabilities. The kind of buffer considered in [11], as well as here, is made of a number of bi-directional containers (where only a finite subset is actually non-empty), one for each possible name. A bi-directional buffer consists of two distinct buffers, one containing the messages received from the client and available to the server, and the other containing the messages received from the server and available to the client.

In [11] orchestrators have bounded buffering capabilities and that restriction is essential to establish several properties concerning contract orchestrators. In our setting, instead, we eliminate that restriction, so will allow more client/server pairs to be compliant, like for instance the pair $\mathsf{rec}\,x.a.x$ and $\mathsf{rec}\,x.\overline{b}.\overline{a}.x$, and those mentioned in the example in the introduction.

The formalisation of the notion of buffer is hence as follows.

Definition 4.1 *(Buffers).*

1. A bi-directional buffer $\mathbb{B}$ is a set of the form $\{\,{}^{c_a}a^{s_a} \mid a \in \mathcal{N}\,\}$ where, for any $a \in \mathcal{N}$, $c_a, s_a \in \mathbb{Z}$.
   This notation can be intuitively interpreted as follows:
   - The number c_a in ${}^{c_a}a^{s_a}$ represents how many as are in the part of the buffer containing messages which can be delivered to the server. An orchestrator modifies that part of its buffer by means of actions of the form $\langle a, \varepsilon\rangle$.
   - The number s_a in ${}^{c_a}a^{s_a}$ represents how many as are in that part of the buffer that contains messages which can be delivered to the client. An orchestrator modifies that part of its buffer by means of actions of the form $\langle \varepsilon, a\rangle$.

   Notice that the numbers c_a and s_a can be negative in order to represent attempts of extracting an a from a buffer containing no a at all.
2. We define: $\tilde{\emptyset} = \{\,{}^{0}a^{0} \mid a \in \mathcal{N}\,\}$ and

$$\begin{aligned}\lfloor^{+}_{a}\mathbb{B} &= (\mathbb{B} \setminus \{\,{}^{c_a}a^{s_a}\,\}) \cup \{\,{}^{c_a+1}a^{s_a}\,\}\\ \lfloor^{-}_{a}\mathbb{B} &= (\mathbb{B} \setminus \{\,{}^{c_a}a^{s_a}\,\}) \cup \{\,{}^{c_a-1}a^{s_a}\,\}\end{aligned}$$

$$\begin{aligned}\mathbb{B}^{+}_{a}\rfloor &= (\mathbb{B} \setminus \{\,{}^{c_a}a^{s_a}\,\}) \cup \{\,{}^{c_a}a^{s_a+1}\,\}\\ \mathbb{B}^{-}_{a}\rfloor &= (\mathbb{B} \setminus \{\,{}^{c_a}a^{s_a}\,\}) \cup \{\,{}^{c_a}a^{s_a-1}\,\}\end{aligned}$$

3. We denote with $|\mathbb{B}|_a$ the number of as in the server-to-client part of the buffer, i.e. $|\mathbb{B}|_a = s_a$ and similarly for the client-to-server part, i.e. ${}_a|\mathbb{B}| = c_a$.
4. $\mathbb{B}\mu$, the state of a buffer $\mathbb{B}$ after an orchestration action μ, is defined by

$$\begin{aligned}\mathbb{B}\langle\overline{a},\varepsilon\rangle &= \lfloor^{-}_{a}\mathbb{B}\\ \mathbb{B}\langle a,\varepsilon\rangle &= \lfloor^{+}_{a}\mathbb{B}\end{aligned}\qquad \mathbb{B}\langle\alpha,\overline{\alpha}\rangle = \mathbb{B}\qquad \begin{aligned}\mathbb{B}\langle\varepsilon,\overline{a}\rangle &= \mathbb{B}^{-}_{a}\rfloor\\ \mathbb{B}\langle\varepsilon,a\rangle &= \mathbb{B}^{+}_{a}\rfloor\end{aligned}$$

5. By $\mathbb{B}\boldsymbol{\mu}$ we denote the buffer $\mathbb{B}$ after the sequence $\boldsymbol{\mu}$ of orchestration actions.

Before proceeding with the definition of respectfulness, we have to define the notion of left-restriction of a sequence with respect to a name.

Definition 4.2. Given $\boldsymbol{\mu} \in \mathbf{OrchAct}^{*} \cup \mathbf{OrchAct}^{\infty}$ and $a \in \mathcal{N}$, we define ${}_a\downarrow\boldsymbol{\mu}$, the left-restriction of $\boldsymbol{\mu}$ to the name a, as follows (λ is the empty sequence):

$$\begin{aligned}{}_a\downarrow\lambda &= \lambda,\\ {}_a\downarrow(\mu\boldsymbol{\mu'}) &= \mu\,{}_a\downarrow\boldsymbol{\mu'}, && \text{if } \mu \in \{\langle\varepsilon,\overline{a}\rangle, \langle a,\varepsilon\rangle\},\\ {}_a\downarrow(\mu\boldsymbol{\mu'}) &= {}_a\downarrow\boldsymbol{\mu'} && \text{otherwise.}\end{aligned}$$

The notion of respectfulness for orchestrators is now formally defined in terms of respectfulness of its traces.

Definition 4.3 *(Respectful sequences and orchestrators).* Let $\boldsymbol{\mu} \in \mathbf{OrchAct}^* \cup \mathbf{OrchAct}^\infty$ and $\mu \in \mathbf{OrchAct}$.

1. For $S \subseteq \mathbf{OrchAct}$, we say that $\boldsymbol{\mu}$ is *definitely-S* whenever:
$$\exists k\, \forall m \geq k\, [\, \boldsymbol{\mu}|_i \in S\,];$$
where $\boldsymbol{\mu} = \mu_1 \cdots \mu_n$, then $\boldsymbol{\mu}|_i = \mu_i$.
For sets that are singletons we write 'definitely-μ' instead of 'definitely-$\{\mu\}$.'
2. We say that $\boldsymbol{\mu}$ is *sound* whenever:
$$\forall a \in \mathcal{N}\, \forall n \leq |\boldsymbol{\mu}|\, [\, {}_a|\tilde{\emptyset}\mu_1 \cdots \mu_n| \geq 0 \text{ and } |\tilde{\emptyset}\mu_1 \cdots \mu_n|_a \geq 0\,]$$
where $\boldsymbol{\mu} = \mu_1 \cdots \mu_n$.
Recall that, as defined in Definition 4.1(2), $\tilde{\emptyset}$ denotes the empty buffer.
3. We say that $\boldsymbol{\mu}$ is a *client-respectful* sequence whenever, for any $a \in \mathcal{N}$:
$${}_a\downarrow\boldsymbol{\mu} \text{ is finite and } {}_a|\tilde{\emptyset}\boldsymbol{\mu}| = 0 \quad \text{or} \quad {}_a\downarrow\boldsymbol{\mu} \text{ is infinite and non-definitely-}\langle a, \varepsilon\rangle$$
4. We say that $\boldsymbol{\mu}$ is *non-definitely server-inputted* whenever:
$$\boldsymbol{\mu} \text{ is infinite} \implies \boldsymbol{\mu} \text{ is non-definitely-}\{\, \langle \varepsilon, a\rangle \mid a \in \mathcal{N}\,\}$$
5. We say that $\boldsymbol{\mu}$ is *respectful* whenever $\boldsymbol{\mu}$ is sound, client-respectful and non-definitely server-inputted.
6. We say that an orchestrator f is *respectful* whenever every $\boldsymbol{\mu} \in \mathsf{MaxTr}(f)$ is so.

We will look now at some examples in order to get a better intuition about the above definition.

Example 4.4.

- The finite sequence $\langle a, \varepsilon\rangle.\langle \varepsilon, \overline{b}\rangle.\langle \varepsilon, \overline{a}\rangle$ is not respectful since it is not sound. In fact, for the name b, we have that
$$|\tilde{\emptyset}\langle a, \varepsilon\rangle\langle \overline{b}, \varepsilon\rangle|_b = -1 < 0.$$
- The sequence $\langle a, \varepsilon\rangle.\langle b, \varepsilon\rangle.\langle \varepsilon, \overline{a}\rangle$ instead, is sound, but nonetheless it is not client-respectful, since it is not infinite and for the name b we have that
$${}_b|\tilde{\emptyset}\langle a, \varepsilon\rangle\langle b, \varepsilon\rangle\langle \varepsilon, \overline{a}\rangle| = 1 \neq 0.$$
- The orchestrator $f = \langle c, \overline{c}\rangle.\mathsf{rec}\, x.(\langle \overline{a}, a\rangle \vee \langle c, \varepsilon\rangle.\langle b, \overline{b}\rangle.x)$ is not respectful since it is not client-respectful. In fact, for the sequence
$$\boldsymbol{\mu} \;=\; \langle c, \overline{c}\rangle\langle c, \varepsilon\rangle\langle b, \overline{b}\rangle\langle c, \varepsilon\rangle\langle b, \overline{b}\rangle \cdots \;\in\; \mathsf{MaxTr}(f)$$
and for the name c, we have that
$${}_c\downarrow\boldsymbol{\mu} \text{ is infinite and } {}_c\downarrow\boldsymbol{\mu} = \langle c, \varepsilon\rangle\langle c, \varepsilon\rangle\langle c, \varepsilon\rangle \cdots \text{ is definitely-}\langle c, \varepsilon\rangle.$$
In fact, from the very first element on, it is made of $\langle c, \varepsilon\rangle$ actions.
- The orchestrator $f = \langle c, \overline{c}\rangle.\mathsf{rec}\, x.(\langle \overline{a}, a\rangle \vee \langle \varepsilon, b\rangle.\langle \varepsilon, c\rangle.x)$ is not respectful since it is definitely server-inputted. In fact, the infinite sequence
$$\boldsymbol{\mu} \;=\; \langle c, \overline{c}\rangle\langle \varepsilon, b\rangle\langle \varepsilon, c\rangle\langle \varepsilon, b\rangle\langle \varepsilon, c\rangle \cdots \;\in\; \mathsf{MaxTr}(f)$$
is definitely-$\{\, \langle \varepsilon, a\rangle \mid a \in \mathcal{N}\,\}$. The orchestrator f in the introduction, instead, is non-definitely server-inputted, and also respectful, as a matter of fact.

Remark 4.5. The sequence ${}_a\downarrow\boldsymbol{\mu}$, used to formalise the notion of client-respectfulness in Definition 4.3(3) above, is defined in such a way it does not take into account synchronous orchestration actions like $\langle a, \overline{a}\rangle$ possibly present in $\boldsymbol{\mu}$ (see Definition 4.2).

By defining ${}_a\downarrow\boldsymbol{\mu}$ in such a way, we manage to rule out orchestrators like
$$g = \langle a, \varepsilon\rangle.\mathsf{rec}\, x.\langle a, \overline{a}\rangle.x$$
Intuitively, g has to be ruled out since the first a coming from the client will never be delivered to the server since any subsequent output $\overline{a}$ will be paired with a further input of a. Formally, g is not client-respectful, and hence it is not respectful at all. In fact, for the sequence $\boldsymbol{\mu} = \langle a, \varepsilon\rangle.\langle a, \overline{a}\rangle.\langle a, \overline{a}\rangle.\langle a, \overline{a}\rangle \ldots \in \mathsf{MaxTr}(g)$ we have that ${}_a\downarrow\boldsymbol{\mu} = \langle a, \varepsilon\rangle$ and ${}_a|\tilde{\emptyset}\boldsymbol{\mu}| \neq 0$.

Ruling out orchestrators like g might be irrelevant when we take into account contexts where distinct occurrences of the same message are indistinguishable. In general, however, the number of input–output actions matters and we wish to maintain our formalism as general as possible.

As noticed in Remark 2.4, if we consider a synchronous action $\langle a, \overline{a}\rangle$ as syntactic sugar for the pair of asynchronous actions $\langle a, \varepsilon\rangle.\langle \varepsilon, \overline{a}\rangle$, the orchestrator above would actually be client-respectful. In fact, it would have only the maximal trace

$$\boldsymbol{\mu'} = \langle a, \varepsilon\rangle.\langle a, \varepsilon\rangle.\langle \varepsilon, \overline{a}\rangle.\langle a, \varepsilon\rangle.\langle \varepsilon, \overline{a}\rangle.\langle a, \varepsilon\rangle.\langle \varepsilon, \overline{a}\rangle \ldots$$

where ${}_a\!\downarrow\boldsymbol{\mu'} = \boldsymbol{\mu'}$ is infinite and non-definitely-$\langle a, \varepsilon\rangle$. This means that, in order to maintain our formalism as general as possible, the presence of synchronous actions does matter.

We can now define our proper notion of orchestrated compliance for session contracts.

Definition 4.6 *(Orchestrated session compliance).* We say that a client ρ *is compliant with a server* σ *through the orchestration of* f, and denote this by $f : \rho \dashv\!\vdash \sigma$, whenever

$$f : \rho \dashv\!\vdash^{\mathsf{d}} \sigma \text{ and } f \text{ is respectful.}$$

We say that ρ *is compliant with* σ, written $\rho \dashv\!\vdash \sigma$, if $f : \rho \dashv\!\vdash \sigma$ for some f.

4.1. Orchestrated compliance decidability

We proved $f : \rho \dashv\!\vdash^{\mathsf{d}} \sigma$ to be decidable in Corollary 3.8. Hence, by Definition 4.6, decidability of $f : \rho \dashv\!\vdash \sigma$ immediately follows from decidability of the property of respectfulness for orchestrators. This can be obtained by means of a characterisation based on the notion of buffer-aware trees and its related labellings whose definition and technicalities will be provided in Appendix B.

Theorem 4.7 *(Orchestrator-respectfulness decidability). Orchestrator respectfulness is decidable.*

Proof. See Appendix B □

Corollary 4.8. *For any* f, ρ, *and* σ, $f : \rho \dashv\!\vdash \sigma$ *is decidable.*

Proof. Immediate from Corollary 3.8 and Theorem 4.7. □

5. Orchestrated subcontract relations

The notion of compliance naturally induces a substitutability relation on servers that may be used for implementing contract-based query engines (see [11] for a detailed discussion). We investigate now the session sub-contract relations induced by our orchestrated compliance on session contracts.

The following definition (Definition 5.1) uses the concept of compliance without orchestrators ($\dashv$), which is called here *strong compliance* (see e.g. [9,7] for more details and references to the literature). Let $\rho \parallel \sigma \longrightarrow \rho' \parallel \sigma'$ be defined as in CCS; then $\rho \dashv \sigma$ if and only if for all ρ', σ'

$$\rho \parallel \sigma \longrightarrow^{*} \rho' \parallel \sigma' \not\longrightarrow \implies \rho' = \mathbf{1}.$$

Now, by adapting [11] to the present context, we define:

Definition 5.1 *(Orchestrated subcontract relations).* Let $\sigma, \sigma' \in \mathsf{SC}$ and $f \in \mathsf{Orch}$. We define

$$\sigma \overset{\circ}{\preccurlyeq}_f \sigma' \triangleq \forall \rho\, [\rho \dashv \sigma \implies f : \rho \dashv\!\vdash \sigma']$$
$$\sigma \overset{\circ}{\preccurlyeq} \sigma' \triangleq \exists f\, [\sigma \overset{\circ}{\preccurlyeq}_f \sigma']$$

Remark 5.2. Note that by Definition 5.1, $\sigma \overset{\circ}{\preccurlyeq} \sigma'$ holds whenever there exist *one* orchestrator f such that $\sigma \overset{\circ}{\preccurlyeq}_f \sigma'$, namely $f : \rho \dashv\!\vdash \sigma'$ for any ρ that is strongly compliant with σ. That is: f does not depends on the particular client ρ. This is not restrictive, and can be equivalently relaxed to

$$\sigma \overset{\circ}{\preccurlyeq} \sigma' \triangleq \forall \rho\, [\rho \dashv \sigma \implies \rho \dashv\!\vdash \sigma']$$

as will be established in Corollary 5.10.

Remark 5.3. One could wonder whether, by using the notion of strong compliance in our definition of $\overset{\circ}{\preccurlyeq}_f$, we are actually formalising a rather restricted notion of service substitutability. As a matter of fact, this is not the case: by knowing $\sigma \overset{\circ}{\preccurlyeq}_f \sigma'$, we are not only assured (by definition) that any client with which σ is strongly compliant is also satisfied by σ' through the mediation of f, but, as mentioned in the Introduction, we are also assured that, if a client with which σ is orchestrated-compliant by means of some f', then that client can be satisfied by σ' as well, through the mediation of a suitable (computable) orchestrator depending on f and f'.

In particular, if we have $\sigma \overset{\circ}{\preccurlyeq}_f \sigma'$ and $f' : \rho \dashv\!\vdash \sigma$ holds for a client ρ, it is possible to show (see Corollary 5.7 below) that $f' \bullet f : \rho \dashv\!\vdash \sigma'$, where $\cdot \bullet \cdot$ is the computable composition operation between orchestrators that we shall formalise in Definition D.2. So the relation $\overset{\circ}{\preccurlyeq}_f$ can be safely used, for instance, to update the services in a repository in order to widen the range of the clients which can use it.

The following alternative relation $\overset{\circ}{\sqsubseteq}_f$ could look more natural at first sight

$$\sigma \overset{\circ}{\sqsubseteq}_f \sigma' \triangleq \forall \rho \, [\, f : \rho \dashv\!\vdash \sigma \implies f : \rho \dashv\!\vdash \sigma' \,]$$

but it would correspond to a not general enough notion of substitutability. By knowing $\sigma \overset{\circ}{\sqsubseteq}_f \sigma'$ we would be assured that the service σ can be safely replaced by the service σ' only in case we restrict the clients of our service to those with which σ is orchestrated-compliant through the mediation of the particular orchestrator f.

We now prove a property which is of use to establish several results.

Proposition 5.4 *(Main Property).*

1. *There exists a* respectfulness-preserving *operator* $\cdot \bullet \cdot$ *on orchestrators, such that*
$$f : \rho \dashv\!\vdash \sigma \;\&\; g : \overline{\sigma} \dashv\!\vdash \sigma' \implies f \bullet g : \rho \dashv\!\vdash \sigma'.$$
2. $\rho \dashv\!\vdash \sigma \;\&\; \overline{\sigma} \dashv\!\vdash \sigma' \implies \rho \dashv\!\vdash \sigma'$.
3. $\rho \dashv \sigma \;\&\; f : \overline{\sigma} \dashv\!\vdash \sigma' \implies f : \rho \dashv\!\vdash \sigma'$.

Proof. See Appendix D. □

The definition of $f \bullet g$ is in D.4 and the proof of the Main Property Proposition above is deferred to Appendix D because of its complexity and length. As far as strong compliance is concerned, the proof of the analogous of the Main Property is relatively simple. In the present context, however, the difficulty of part (2) is due to the fact that, given an orchestrator f for $\rho \dashv\!\vdash \sigma$ and an orchestrator g for $\overline{\sigma} \dashv\!\vdash \sigma'$, we have to find a third orchestrator, in principle a new one, for $\rho \dashv\!\vdash \sigma'$. Moreover, we have to show that orchestrator is respectful. Property (3) follows from (2) in a non-trivial way.

Lemma 5.5. *For any* σ,

1. $\overline{\sigma} \dashv \sigma$.
2. $\overline{\sigma} \dashv\!\vdash \sigma$.

Proof.

1. Since $\overline{\sigma}$ is obtained from σ by exchanging each a with $\overline{a}$ and $+$ with $\oplus$, this part is an immediate consequence of CCS communication rules used in the reduction of $\overline{\sigma} \parallel \sigma$.
2. By definition of $\overline{\sigma} \dashv\!\vdash \sigma$ we have to show that $f : \overline{\sigma} \dashv\!\vdash^{\mathsf{d}} \sigma$ for some respectful f. The following rule and its symmetric version are instances of rules (CPLΣ-$\oplus$) and (CPL$\oplus$-Σ) in Fig. 2 respectively:
$$\frac{\Gamma, \Sigma_{i\in I} a_i.\overline{\sigma}_i \dashv^{\mathsf{d}} \oplus_{i\in I} \overline{a}_i.\sigma_i \rhd f_j : \overline{\sigma}_j \dashv^{\mathsf{d}} \sigma_j \quad (\forall j \in I)}{\Gamma \rhd \bigvee_{i\in I} \langle \overline{a}_i, a_i \rangle . f_i : \Sigma_{i\in I} a_i.\overline{\sigma}_i \dashv^{\mathsf{d}} \oplus_{i\in I} \overline{a}_i.\sigma_i} \text{(CPL}\Sigma\text{-}\oplus')$$
By definition of dual contracts, a proof system using only (Ax), (Hyp), (CPLΣ-$\oplus'$) and (CPL$\oplus$-Σ') suffices to derive an orchestrator f for $\overline{\sigma}$ and σ. But then by construction of f, it is synchronous, hence it is trivially respectful. □

From part (3) of the Main Property, we can characterise $\overset{\circ}{\preccurlyeq}_f$ and show its decidability in a relatively simple way. Actually, the following corollary is a classic result when compliance and duality are computable.

Corollary 5.6 *(Decidability of $\overset{o}{\preccurlyeq}_f$). For any $\sigma, \sigma' \in \mathsf{SC}$ and $f \in \mathsf{Orch}$:*

1. $\sigma \overset{o}{\preccurlyeq}_f \sigma' \iff f : \overline{\sigma} \dashv\!\vdash \sigma'$
2. *$\sigma \overset{o}{\preccurlyeq}_f \sigma'$ is decidable.*

Proof.

1. ($\Rightarrow$): By contraposition, assume that $f : \overline{\sigma} \not\dashv\!\vdash \sigma'$. Since $\overline{\sigma} \dashv \sigma$, then by definition of $\overset{o}{\preccurlyeq}_f$ we have $\sigma \overset{o}{\not\preccurlyeq}_f \sigma'$.
   ($\Leftarrow$): Let $f : \overline{\sigma} \dashv\!\vdash \sigma'$. If $\rho \dashv \sigma$, then from $f : \overline{\sigma} \dashv\!\vdash \sigma'$, we get $f : \rho \dashv\!\vdash \sigma'$ by Proposition 5.4 (3), and therefore $\sigma \overset{o}{\preccurlyeq}_f \sigma'$ by definition.
2. From (1) and decidability of $f : \overline{\sigma} \dashv\!\vdash \sigma'$. □

As mentioned in Remark 5.3, we can show that even if strong compliance is used in the definition of $\overset{o}{\preccurlyeq}_f$, the latter relation embodies a quite general notion of substitutability, also when clients are orchestrated.

Corollary 5.7 *(Substitutability embodiment). Let $\sigma \overset{o}{\preccurlyeq}_g \sigma'$ and let $f : \rho \dashv\!\vdash \sigma$. Then $f \bullet g : \rho \dashv\!\vdash \sigma'$.*

Proof. By Corollary 5.6(1), from $\sigma \overset{o}{\preccurlyeq}_g \sigma'$ we get $g : \overline{\sigma} \dashv\!\vdash \sigma'$. Hence, from $f : \rho \dashv\!\vdash \sigma$ and Proposition 5.4(1) we get the thesis. □

As a further consequence of part (3) of the Main Property we have that, given a client ρ and an orchestrator f, its dual $\overline{\rho}$ precedes all the servers of ρ that are orchestrated by f (notice that it could be the case that $f : \rho \not\dashv\!\vdash \overline{\rho}$).

Corollary 5.8 *(Dual precedes all servers). Let $\rho \in \mathsf{SC}$ and $f \in \mathsf{Orch}$. Then, for any $\sigma \in \mathsf{SC}$:*

$$f : \rho \dashv\!\vdash \sigma \implies \overline{\rho} \overset{o}{\preccurlyeq}_f \sigma$$

Proof. Suppose that $f : \rho \dashv\!\vdash \sigma$ and $\tau \dashv \overline{\rho}$. Since $\overline{\overline{\rho}} = \rho$, by Proposition 5.4 (3) we know that $f : \tau \dashv\!\vdash \sigma$; hence $\overline{\rho} \overset{o}{\preccurlyeq}_f \sigma$ by definition. □

A parameterised transitivity property holds for $\overset{o}{\preccurlyeq}_f$:

Corollary 5.9 *(Transitivity of $\overset{o}{\preccurlyeq}_f$).* $\rho \overset{o}{\preccurlyeq}_f \delta \;\&\; \delta \overset{o}{\preccurlyeq}_g \sigma \implies \rho \overset{o}{\preccurlyeq}_{f \bullet g} \sigma$.

Proof. Let $\rho \overset{o}{\preccurlyeq}_f \delta$ and $\delta \overset{o}{\preccurlyeq}_g \sigma$. By Corollary 5.6 (1) we have that $f : \overline{\rho} \dashv\!\vdash \delta$ and $g : \overline{\delta} \dashv\!\vdash \sigma$. Using Proposition 5.4 (1) we get $f \bullet g : \overline{\rho} \dashv\!\vdash \sigma$, so that by Corollary 5.6 (1) we conclude that $\rho \overset{o}{\preccurlyeq}_{f \bullet g} \sigma$. □

From part (3) of the Main Property we also deduce that the alternative definition of $\overset{o}{\preccurlyeq}$ mentioned in Remark 5.2 is actually equivalent to the second one provided in Definition 5.1.

Corollary 5.10 *(Equivalent definitions of $\overset{o}{\preccurlyeq}$). For any σ, σ', the following two conditions are equivalent.*

1. $\forall \rho \, [\, \rho \dashv \sigma \implies \rho \dashv\!\vdash \sigma' \,]$.
2. $\exists f \, \forall \rho \, [\, \rho \dashv \sigma \implies f : \rho \dashv\!\vdash \sigma' \,]$.

Proof. Since $\rho \dashv\!\vdash \sigma'$ is $f : \rho \dashv\!\vdash \sigma'$ for some f, while $\rho \dashv \sigma$ does not depend on f, condition (1) is equivalent to

$$\forall \rho \, \exists f \, [\, \rho \dashv \sigma \implies f : \rho \dashv\!\vdash \sigma' \,].$$

On the other hand (2) is just

$$\exists f \, \forall \rho \, [\, \rho \dashv \sigma \implies f : \rho \dashv\!\vdash \sigma' \,],$$

hence we have that (2) implies (1) by logic.

Vice versa, for an arbitrary ρ suppose that $\rho \dashv \sigma$. By 5.5(2) we have $\overline{\sigma} \dashv\!\vdash \sigma$, i.e. there exists an f such that $f : \overline{\sigma} \dashv\!\vdash \sigma$, which clearly does not depend on ρ. Hence we have that $f : \rho \dashv\!\vdash \sigma$ by Proposition 5.4(3), and therefore we conclude that (1) implies (2). □

Corollary 5.11. $\dashv \subsetneq \dashv\!\vdash$.

Proof. Let $\rho \dashv \sigma$. By 5.5(2) we have that $f : \overline{\sigma} \dashv\!\shortmid \sigma$ for a certain f. We immediately get $f : \sigma \dashv\!\shortmid \sigma$ by 5.4(3). This proves $\subseteq$. In order to get $\subsetneq$, it is immediate to check that $\overline{a}.\overline{b} \not\dashv b.a$, whereas $f : \overline{a}.\overline{b} \dashv\!\shortmid b.a$ for the respectful orchestrator $f = \langle a, \varepsilon\rangle.\langle b, \overline{b}\rangle.\langle \varepsilon, \overline{a}\rangle$. □

By means of part (2) of the Main Property all the results shown for $\overset{\circ}{\preccurlyeq}_f$ can be proven for $\overset{\circ}{\preccurlyeq}$ as well, with the exception of the decidability property, for which we cannot rely on decidability of $\dashv\!\shortmid$.

Corollary 5.12 *(Properties of $\overset{\circ}{\preccurlyeq}$).*

1. $\sigma \overset{\circ}{\preccurlyeq} \sigma'$ *iff* $\overline{\sigma} \dashv\!\shortmid \sigma'$.
2. $\overset{\circ}{\preccurlyeq}$ *is decidable iff* $\dashv\!\shortmid$ *is decidable.*
3. *Let* $\rho \in \mathsf{SC}$. *Then* $\overline{\rho}$ *is a least element among* ρ*'s servers, that is, for every* σ*:* $\rho \dashv\!\shortmid \sigma \implies \overline{\rho} \overset{\circ}{\preccurlyeq} \sigma$.
4. $\overset{\circ}{\preccurlyeq}$ *is a transitive relation.*

Proof.

1. ($\Rightarrow$): By contraposition, assume $\overline{\sigma} \not\dashv\!\shortmid \sigma'$. From this we derive $\sigma \not\overset{\circ}{\preccurlyeq} \sigma'$ by contradiction. In fact, assuming $\sigma \overset{\circ}{\preccurlyeq} \sigma'$ immediately gives a contradiction by definition of $\overset{\circ}{\preccurlyeq}$, since we have that $\sigma \dashv \overline{\sigma}$ (Lemma 5.5) but $\overline{\sigma} \not\dashv\!\shortmid \sigma'$.
   ($\Leftarrow$): Let $\overline{\sigma} \dashv\!\shortmid \sigma'$. In order to show $\sigma \overset{\circ}{\preccurlyeq} \sigma'$, by definition, let $\rho \dashv \sigma$. Then, from $\overline{\sigma} \dashv\!\shortmid \sigma'$, we get $\rho \dashv\!\shortmid \sigma'$ by Proposition 5.4.
2. Immediate from (1).
3. Assume $\rho \dashv\!\shortmid \sigma$. By definition, in order to show $\overline{\rho} \overset{\circ}{\preccurlyeq} \sigma$, assume $\tau \dashv \overline{\rho}$. From Proposition 5.4(2) we get $\tau \dashv\!\shortmid \sigma$.
4. We first observe that $\overline{\rho}$ belongs to the set of ρ's servers, by Lemma 5.5(2). In order to show that it is a least element, let $\rho \overset{\circ}{\preccurlyeq} \delta$ and $\delta \overset{\circ}{\preccurlyeq} \sigma$. Hence, by (1) we have that $\overline{\rho} \dashv\!\shortmid \delta$ and $\overline{\delta} \dashv\!\shortmid \sigma$. Then $\overline{\rho} \dashv\!\shortmid \sigma$ by Proposition 5.4(2). So, again by (1), we get $\rho \overset{\circ}{\preccurlyeq} \sigma$. □

Remark 5.13. Notice that in 5.12(3) we said the $\overline{\rho}$ is *a* least element, since $\overset{\circ}{\preccurlyeq}$ is a not antisymmetric preorder. In order to show $\overset{\circ}{\preccurlyeq}$ not to be antisymmetric, a simple counterexample can be used. We have that $ab \overset{\circ}{\preccurlyeq} ba$ and $ba \overset{\circ}{\preccurlyeq} ab$, but $ab \neq ba$.

6. Orchestrators for `skp`

As discussed in the Introduction, we prove now the equivalence between the relation of `skp`-compliance as introduced in [13] and a restricted version of our session orchestrated compliance, `skp-Orch`-compliance.

It will be possible to devise also a synthesis algorithm finding a suitable orchestrator for a client ρ and a server σ in case they are `skp-Orch`-compliant. It will turn out that the orchestrator mediating the interactions between ρ and server σ is a byproduct of part of the decision algorithm for the relation `skp-Orch`-compliance.

We begin by recalling the formal definition of `skp`-compliance for session contracts.

6.1. The relation of `skp`-compliance [13]

Definition 6.1 *(`skp`-LTS for client–server pairs [13]).* We write $\rho \not\Downarrow \alpha$ for $\neg\exists\, \rho'\ [\rho \overset{\alpha}{\Longrightarrow} \rho']$.

Let **SkipAct** $= \{\tau, \mathtt{skp}\}$ be the set of the synchronisation actions and $\rho \parallel \sigma$ denote, as usual, the parallel composition of session behaviours in SC; we define the following LTS formalising synchronous and skipping communications:

$$\frac{\rho \xrightarrow{\rho}{}'}{\rho \parallel \sigma \rightarrow_{\mathsf{s}} \rho' \parallel \sigma} \qquad \frac{\sigma \xrightarrow{\sigma}{}'}{\rho \parallel \sigma \rightarrow_{\mathsf{s}} \rho \parallel \sigma'}$$

$$\frac{\rho \not\Downarrow a \quad \sigma \xrightarrow{\overline{a}} \sigma'}{\rho \parallel \sigma \xrightarrow{\mathtt{skp}}_{\mathsf{s}} \rho \parallel \sigma'} \qquad \frac{\rho \xrightarrow{\alpha} \rho' \quad \sigma \xrightarrow{\overline{\alpha}} \sigma'}{\rho \parallel \sigma \xrightarrow{\tau}_{\mathsf{s}} \rho' \parallel \sigma'}$$

where $a \in \mathcal{N}$ (and hence $\overline{a} \in \overline{\mathcal{N}}$) and where $\alpha \in \mathcal{N} \cup \overline{\mathcal{N}}$.

The reason behind allowing clients to skip some actions on the server side is to let more clients to synchronise with servers that essentially provide the required service but for some supplementary (and possibly redundant) information.

As usual, we abbreviate $\Rightarrow_{\mathsf{s}} = \rightarrow_{\mathsf{s}}{}^*$ and $\overset{\xi}{\Rightarrow}_{\mathsf{s}} = \Rightarrow_{\mathsf{s}} \circ \xrightarrow{\xi}_{\mathsf{s}} \circ \Rightarrow_{\mathsf{s}}$, where $\xi \in$ **SkipAct**.

Definition 6.2 *(Synchronisation `skp`-traces [13]).* The mapping

$$\mathsf{skTr} : \mathsf{SC} \times \mathsf{SC} \rightarrow ((\mathbf{SkipAct} \cup \{\checkmark\})^* \cup \mathbf{SkipAct}^{\infty})$$

is defined by

$$\mathsf{skTr}(\rho \parallel \sigma) = \begin{cases} \{\checkmark\} & \text{if } \rho = \mathbf{1} \\ \{\xi\ \boldsymbol{\chi} \mid \rho \parallel \sigma \stackrel{\xi}{\Longrightarrow}_{\mathrm{s}} \rho' \parallel \sigma' \ \&\ \boldsymbol{\chi} \in \mathsf{skTr}(\rho' \parallel \sigma')\} & \text{if } \exists \zeta \in \mathbf{SkipAct}\,[\,\rho \parallel \sigma \stackrel{\zeta}{\Longrightarrow}_{\mathrm{s}}\,] \\ \{\varepsilon\} & \text{otherwise} \end{cases}$$

Let $\boldsymbol{\xi} \in \mathbf{SkipAct}^{\infty}$ with $\boldsymbol{\xi} = \xi_1\xi_2\cdots$. We say that $\boldsymbol{\xi}$ is *definitely*-`skp` whenever all ξ_h are equal to `skp` after a certain ξ_k: $\exists k\,\forall h > k\ [\,\xi_h = \mathtt{skp}\,]$.

The notion of `skp`-compliance can now be formalised in terms of synchronisation traces as follows.

Definition 6.3 (`skp`-*compliance [13]*). The *client* ρ is `skp`-compliant with the *server* σ, written $\rho \dashv^{\mathtt{skp}} \sigma$, whenever, for any $\boldsymbol{\xi} \in \mathsf{skTr}(\rho \parallel \sigma)$ either $\boldsymbol{\xi} = \boldsymbol{\xi}'\checkmark$ or $\boldsymbol{\xi}$ is infinite and not definitely-`skp`.

Theorem 6.4 *([13]). The relation $\dashv^{\mathtt{skp}}$ is decidable.*

6.2. Orchestrated compliance with skipping orchestrators

We introduce now a restriction of $\dashv\!\vdash$ equivalent to $\dashv^{\mathtt{skp}}$.

A skipping orchestrator is an orchestrator possessing just the server-to-client buffer, which can be used just for discarding server's outputs.

Definition 6.5 *(Skipping orchestrators).*

1. We define

$$\mathbf{SkpOrchAct} = \{\langle \alpha, \overline{\alpha}\rangle \mid \alpha \in \mathcal{N} \cup \overline{\mathcal{N}}\} \cup \{\langle \varepsilon, a\rangle \mid a \in \mathcal{N}\}.$$

2. We define the set SkpOrch of *skipping orchestrators* as follows: $f \in \mathsf{SkpOrch}$ whenever $f \in \mathsf{Orch}$ and the set of orchestration actions used in f is restricted to **SkpOrchAct**.

By the structure of skipping orchestrators, we immediately get the following:

Fact 6.6. *Let $f \in \mathsf{SkpOrch}$. Then f is sound and client-respectful.*

Of course a skipping orchestrator could be definitely server-inputted, like for example $\langle \overline{a}, a\rangle.\mathsf{rec}\,x.\langle \varepsilon, b\rangle$.

In the following we shall use 'ndsi' for 'non-definitely server-inputted'.

Remark 6.7. Notice that the existence of a ndsi skipping orchestrator f such that $f : \rho \dashv\!\vdash^{\mathsf{d}} \sigma$ is a necessary but not sufficient condition for $\rho \dashv^{\mathtt{skp}} \sigma$ to hold.

In fact, the client $\mathsf{rec}\,x.b.a.x$ complies with the server $\overline{b}.\mathsf{rec}\,x.\overline{b}.\overline{a}.x$ by means of the orchestrator $\langle \varepsilon, b\rangle.\mathsf{rec}\,x.\langle \varepsilon, b\rangle.\langle \overline{a}, a\rangle$, which is a ndsi skipping orchestrator, but nonetheless we have that $\mathsf{rec}\,x.b.a.x \not\dashv^{\mathtt{skp}} \overline{b}.\mathsf{rec}\,x.\overline{b}.\overline{a}.x$.

We now introduce the notion of $(\rho \parallel \sigma)$-skipping orchestrator, a skipping orchestrator that does not skip an output action of the server when a corresponding input is present on the client side.

Definition 6.8 $(\rho \parallel \sigma)$-*skipping orchestrators.* Let $f \in \mathsf{SkpOrch}$. We define the set $(\rho \parallel \sigma)$-SkpOrch of $(\rho \parallel \sigma)$-skipping orchestrators as follows: $f \in (\rho \parallel \sigma)$-SkpOrch whenever for any $\rho', \sigma' \in \mathsf{SC}$, $f \in \mathsf{SkpOrch}$, $\boldsymbol{\mu} \in \mathbf{SkipAct}^{*}$ and $a \in \mathcal{N}$,

$$\rho \parallel_f \sigma \stackrel{\boldsymbol{\mu}}{\Longrightarrow} \rho' \parallel_{f'} \sigma \xrightarrow{\langle \varepsilon, a\rangle} \text{ implies } \rho' \not\Downarrow a$$

We can now define the relation `skp-Orch`-compliance as a restriction of our notion of orchestrated compliance.

Definition 6.9 (`skp-Orch`-*compliance).*

1. We say that a client ρ is

`skp-Orch`-compliant with a server σ through the orchestration of f,

and denote this by $f : \rho \dashv\!\vdash^{\mathtt{skp}}_{\mathtt{Orch}} \sigma$, whenever

$\mathbf{SB}(\Gamma, \rho, \sigma) =$

1. **if** $\rho = \mathbf{1}$ **then** 1
2. **else if** $x : \rho \dashv^{\mathsf{d}} \sigma \in \Gamma$ **then** x
3. **else if** $\rho = \oplus_{i\in I}\, \overline{a}_i.\rho_i$ **and** $\sigma = \oplus_{j\in J}\, \overline{a}_j.\sigma_i$ **then**
   let $\Gamma' = \Gamma,\ x{:}\rho \dashv^{\mathsf{d}} \sigma$ **in**
   let $(\forall j \in J)\ \ f_j = \mathbf{SB}(\Gamma', \rho, \sigma_j)$ **in**
   $\mathsf{rec}\, x.\bigvee_{j\in J} \langle \varepsilon, a_j\rangle.f_j$
4. **else if** $\rho = \oplus_{i\in I}\, \overline{a}_i.\rho_i$ **and** $\sigma = \Sigma_{j\in J}\, a_j.\sigma_j$ **and** $I \subseteq J$ **then**
   let $\Gamma' = \Gamma,\ x{:}\rho \dashv^{\mathsf{d}} \sigma$ **in**
   let $(\forall i \in I)\ \ f_i = \mathbf{SB}(\Gamma', \rho_i, \sigma_i)$ **in**
   $\mathsf{rec}\, x.\bigvee_{i\in I} \langle a_i, \overline{a}_i\rangle.f_i$
5. **else if** $\rho = \Sigma_{j\in J}\, a_j.\rho_j$ **and** $\sigma = \oplus_{i\in I}\, \overline{a}_i.\sigma_i$ **then**
   let $\Gamma' = \Gamma,\ x{:}\rho \dashv^{\mathsf{d}} \sigma$ **in**
   let $(\forall k \in (J \cap I))\ \ f_k = \mathbf{SB}(\Gamma', \rho_k, \sigma_k)$
   and $(\forall h \in (I \setminus J))\ \ f_h = \mathbf{SB}(\Gamma', \rho, \sigma_h)$ **in**
   $\mathsf{rec}\, x.((\bigvee_{k\in(J\cap I)} \langle \overline{a}_k, a_k\rangle.f_k) \vee (\bigvee_{h\in(I\setminus J)} \langle \varepsilon, a_h\rangle.f_h))$
6. **else fail**

where every time a variable x is introduced, it is a fresh one.

Fig. 4. The algorithm **SB**.

$$f : \rho \dashv\!\vdash \sigma \ \text{ and } \ f^{\mathbf{s}} \in (\rho \parallel \sigma)\text{-SkpOrch}.$$

where $f^{\mathbf{s}} = \mathbf{A}(f, \rho, \sigma, \emptyset)$ and $\mathbf{A}$ is the algorithm described in the proof of Proposition 2.15.

2. We write $\rho \dashv\!\vdash^{\mathtt{skp}}_{\mathtt{Orch}} \sigma$ if there exists an orchestrator f such that $f : \rho \dashv\!\vdash^{\mathtt{skp}}_{\mathtt{Orch}} \sigma$.

We use $f^{\mathbf{s}}$ in the above definition since, for example, it is natural to expect the client $\overline{a}$ to be skp-Orch-compliant with the server $\overline{b}.a$ by means of the orchestrator $\mathsf{rec}\, x.(\langle \varepsilon, b\rangle.\langle a, \overline{a}\rangle \vee \langle \varepsilon, c\rangle.\langle \varepsilon, c\rangle.x)$, which is not ndsi, but its strict version is. As done before, without loss of generality we focus on strict orchestrators.

By Fact 6.6 the following correspondences between $\dashv\!\vdash^{\mathtt{skp}}_{\mathtt{Orch}}$ and $\dashv\!\vdash^{\mathsf{d}}$ descend immediately.

Fact 6.10.

1. $f : \rho \dashv\!\vdash^{\mathtt{skp}}_{\mathtt{Orch}} \sigma \ \Leftrightarrow\ f : \rho \dashv\!\vdash^{\mathsf{d}} \sigma\ \&\ f \in (\rho \parallel \sigma)-\text{SkpOrch}\ \&\ f\ \textit{ndsi}.$
2. $\rho \dashv\!\vdash^{\mathtt{skp}}_{\mathtt{Orch}} \sigma \ \Leftrightarrow\ \exists f \in (\rho \parallel \sigma)\,[\, f : \rho \dashv\!\vdash^{\mathsf{d}} \sigma\ \textit{with}\ f\ \textit{ndsi}\,].$

It is possible to show that the relations $\dashv^{\mathtt{skp}}$ and $\dashv\!\vdash^{\mathtt{skp}}_{\mathtt{Orch}}$ actually coincide. We postpone the proof to Appendix E.

Theorem 6.11. *For any* $\rho, \sigma \in \mathsf{SC}$,

$$\rho \dashv^{\mathtt{skp}} \sigma \quad \Leftrightarrow \quad \rho \dashv\!\vdash^{\mathtt{skp}}_{\mathtt{Orch}} \sigma$$

Proof. See Appendix E (Propositions E.12 and E.14). □

6.3. An orchestrator-synthesis algorithm for $\dashv\!\vdash^{\mathtt{skp}}_{\mathtt{Orch}}$

From the proof of Theorem 6.11 we extract an algorithm that takes a client ρ and a server σ and synthesises an orchestrator f, if any, such that $f : \rho \dashv\!\vdash^{\mathtt{skp}}_{\mathtt{Orch}} \sigma$. Here we do not identify recursive orchestrators with their expansions.

The algorithm is the following:

$\mathbf{S}(\rho, \sigma) =$ **let** $f = \mathbf{SB}(\emptyset, \rho, \sigma)$
in
if ($f \neq$ **fail and** f is ndsi)
then f
else fail

where the procedure **SB** is described in Fig. 4.

$$
\begin{array}{l}
\textbf{Synth}(\Gamma, \rho, \sigma) = \\
\quad \textbf{if } x : \rho \dashv^{\mathsf{d}} \sigma \in \Gamma \textbf{ then } \{x\} \\
\quad \textbf{else if } \rho = \mathbf{1} \textbf{ then } \{\mathbf{1}\} \\
\quad \textbf{else if } \rho = \Sigma_{i\in I}\, a_i.\rho_i \textbf{ and } \sigma = \Sigma_{j\in J}\, a_j.\sigma_j \textbf{ then} \\
\qquad \textbf{let } \Gamma' = \Gamma,\ x{:}\rho \dashv^{\mathsf{d}} \sigma \textbf{ in} \\
\qquad\quad \bigcup_{i\in I}\ \{\mathsf{rec}\, x.\langle \overline{a}_i, \varepsilon\rangle.f \mid f \in \textbf{Synth}\,(\Gamma', \rho_i, \sigma)\} \cup \\
\qquad\qquad \bigcup_{j\in J} \{\mathsf{rec}\, x.\langle \varepsilon, \overline{a}_j\rangle.f \mid f \in \textbf{Synth}\,(\Gamma', \rho, \sigma_j)\} \\
\quad \textbf{else if } \rho = \oplus_{i\in I}\, \overline{a}_i.\rho_i \textbf{ and } \sigma = \oplus_{j\in J}\, \overline{a}_j.\sigma_i \textbf{ then} \\
\qquad \textbf{let } \Gamma' = \Gamma,\ x{:}\rho \dashv^{\mathsf{d}} \sigma \textbf{ in} \\
\qquad\quad \{\mathsf{rec}\, x.\ \bigvee_{i\in I}\langle a_i, \varepsilon\rangle.f_i \mid f_i \in \textbf{Synth}\,(\Gamma', \rho_i, \sigma)\} \cup \\
\qquad\qquad \{\mathsf{rec}\, x.\bigvee_{j\in J}\langle \varepsilon, a_j\rangle.f_j \mid f_j \in \textbf{Synth}\,(\Gamma', \rho, \sigma_j)\} \\
\quad \textbf{else if } \rho = \oplus_{i\in I}\, \overline{a}_i.\rho_i \textbf{ and } \sigma = \Sigma_{j\in J}\, a_j.\sigma_j \textbf{ then} \\
\qquad \textbf{let } \Gamma' = \Gamma,\ x{:}\rho \dashv^{\mathsf{d}} \sigma \textbf{ in} \\
\qquad\quad \{\mathsf{rec}\, x.(\bigvee_{h\in H}\langle a_h, \varepsilon\rangle.f_h)\ \vee\ (\bigvee_{k\in K}\langle a_k, \overline{a}_k\rangle.f_k) \\
\qquad\qquad \mid I = H \cup K,\ K \subseteq J,\ f_h \in \textbf{Synth}(\Gamma', \rho_h, \sigma),\ f_k \in \textbf{Synth}\,(\Gamma', \rho_k, \sigma_k)\} \\
\qquad\quad \cup\ \bigcup_{j\in J}\{\mathsf{rec}\, x.\langle \varepsilon, \overline{a}_j\rangle.f \mid f \in \textbf{Synth}\,(\Gamma', \rho, \sigma_j\} \\
\quad \textbf{else if } \rho = \Sigma_{i\in I}\, a_i.\rho_i \textbf{ and } \sigma = \oplus_{j\in J}\, \overline{a}_j.\sigma_j \textbf{ then} \\
\qquad \textbf{let } \Gamma' = \Gamma,\ x{:}\rho \dashv^{\mathsf{d}} \sigma \textbf{ in} \\
\qquad\quad \{\mathsf{rec}\, x.(\bigvee_{h\in H}\langle \varepsilon, a_h\rangle.f_h)\ \vee\ (\bigvee_{k\in K}\langle \overline{a}_k, a_k\rangle.f_k) \\
\qquad\qquad \mid J = H \cup K,\ K \subseteq I,\ f_h \in \textbf{Synth}((\Gamma', \rho, \sigma_h),\ f_k \in \textbf{Synth}\,(\Gamma', \rho_k, \sigma_k)\} \\
\qquad\quad \cup\ \bigcup_{i\in I}\{\mathsf{rec}\, x.\langle \overline{a}_i, \varepsilon\rangle.f \mid f \in \textbf{Synth}\,(\Gamma', \rho_i, \sigma)\} \\
\quad \textbf{else } \emptyset
\end{array}
$$

Fig. 5. The algorithm **Synth** of [12].

Theorem 6.12. *The algorithm* **S** *is correct and complete with respect to* $\dashv\!\vdash^{\mathtt{skp}}_{\mathtt{Orch}}$.

Proof. See Appendix E (Corollary E.20). □

Remark 6.13. The proof of the completeness part of Theorem 6.12 provided in Appendix E consists of showing that for any ρ and σ such that $f : \rho \dashv\!\vdash^{\mathtt{skp}}_{\mathtt{Orch}} \sigma$ we have that there exists g such that $\mathbf{SB}(\Gamma, \rho, \sigma) = g \neq \mathbf{fail}$, where g is ndsi.

The algorithm **SB** is a variant of the algorithm **Synth** in [12], which we proposed to decide the relation $\dashv\!\vdash$ and that we recall here in Fig. 5. Differently from the algorithm **S**, which produces just one orchestrator, the algorithm **Synth** returns a set of orchestrators.

The algorithm **Synth** is correct, but not complete for the relation $\dashv\!\vdash$: consider, for instance, the client $\overline{a}.\overline{b}$ and the server $b.a$. It is not difficult to check that

$$\textbf{Synth}(\emptyset, \overline{a}.\overline{b}, b.a) = \{\ \langle a, \varepsilon\rangle.\langle b, \varepsilon\rangle,\ \langle a, \varepsilon\rangle.\langle b, \overline{b}\rangle,\ \langle \varepsilon, \overline{b}\rangle.\langle \varepsilon, \overline{a}\rangle.\langle a, \varepsilon\rangle.\langle b, \varepsilon\rangle,\ \langle \varepsilon, \overline{b}\rangle.\langle a, \overline{a}\rangle.\langle b, \varepsilon\rangle\ \}$$

where none of the returned orchestrators is respectful, even if there exist actually some respectful orchestrators for the given client/server pair, namely $\langle a, \varepsilon\rangle.\langle b, \overline{b}\rangle.\langle \varepsilon, \overline{a}\rangle$ and $\langle a, \varepsilon\rangle.\langle b, \varepsilon\rangle.\langle \varepsilon, \overline{b}\rangle.\langle \varepsilon, \overline{a}\rangle$.

In order to prove **Synth** to be complete for the relation $\dashv\!\vdash$, we should be able to modify the halting conditions in such a way the following properties hold:

If $f : \rho \dashv\!\vdash^{\mathsf{d}} \sigma$ with f sound then there exists a g computed by $\textbf{Synth}(\emptyset, \rho, \sigma)$ which is sound.

and

If $f : \rho \dashv\!\vdash^{\mathsf{d}} \sigma$ with f client-respectful then there exists a g computed by $\textbf{Synth}(\emptyset, \rho, \sigma)$ which is client-respectful.

However, at the moment of writing we do not have any extension of the algorithm **Synth** such that the above claims do hold.

Notice that, by checking the clauses of **SB** and **Synth**, it is possible to show that

$$f = \mathbf{S}(\emptyset, \rho, \sigma) \neq \mathbf{fail} \implies f \in \textbf{Synth}(\emptyset, \rho, \sigma)$$

The opposite implication does not hold, since asynchronous exchanges of messages cannot be dealt with by means of the notion of skp-compliance. A simple counterexample is the following one. It is easy to check that

$$\mathbf{S}(\emptyset, \mathsf{rec}\, x.\overline{a}.\overline{b}.x, \mathsf{rec}\, x.b.a.x) = \mathbf{fail}$$

because

$$\mathbf{SB}(\emptyset, \mathrm{rec}\,x.\overline{a}.\overline{b}.x, \mathrm{rec}\,x.b.a.x) = \mathbf{fail}.$$

In particular, the algorithm fails when calling $\mathbf{SB}(\emptyset, \mathrm{rec}\,x.\overline{a}.\overline{b}.x, \mathrm{rec}\,x.b.a.x)$, since clause (3.) cannot be applied because the condition $I \subseteq J$ does not hold. Hence **fail** is returned. On the other hand, we have that $\mathbf{Synth}(\emptyset, \mathrm{rec}\,x.\overline{a}.\overline{b}.x, \mathrm{rec}\,x.b.a.x)$ does not fail. In particular,

$$\mathbf{Synth}(\emptyset, \mathrm{rec}\,x.\overline{a}.\overline{b}.x, \mathrm{rec}\,x.b.a.x) = \left\{ \begin{array}{l} \mathrm{rec}\,x.\langle a,\varepsilon\rangle.\langle b,\varepsilon\rangle.x,\ \mathrm{rec}\,x.\langle a,\varepsilon\rangle.\langle b,\overline{b}\rangle.\langle \varepsilon,\overline{a}\rangle.x, \\ \mathrm{rec}\,x.\langle \varepsilon,\overline{b}\rangle.\langle a,\varepsilon\rangle.\langle b,\varepsilon\rangle.\langle \varepsilon,\overline{a}\rangle.x,\ \mathrm{rec}\,x.\langle \varepsilon,\overline{b}\rangle.\langle a,\varepsilon\rangle.\langle \varepsilon,\overline{a}\rangle.\langle b,\varepsilon\rangle.x, \\ \mathrm{rec}\,x.\langle \varepsilon,\overline{b}\rangle.\langle a,\overline{a}\rangle.\langle b,\varepsilon\rangle.x,\ \mathrm{rec}\,x.\langle \varepsilon,\overline{b}\rangle.\langle \varepsilon,\overline{a}\rangle.x \end{array} \right\}$$

It is then possible to check that amongst these, the only respectful orchestrator is $\mathrm{rec}\,x.\langle a,\varepsilon\rangle.\langle b,\overline{b}\rangle.\langle \varepsilon,\overline{a}\rangle.x$.

7. Related and future work

The present work has been inspired by [11]. It is worth noticing that the results about the $\overset{\circ}{\preccurlyeq}$ relation proved in Section 5 do not follow from the corresponding ones for the subcontract relation of [11] (let us call it $\preccurlyeq^{\mathsf{P}}$ here), given that the two relations are incomparable. For example, we have $\mathrm{rec}\,x.\overline{a}.x \overset{\circ}{\preccurlyeq} \mathrm{rec}\,x.\overline{c}.\overline{a}.x$, but $\mathrm{rec}\,x.\overline{a}.x \not\preccurlyeq^{\mathsf{P}} \mathrm{rec}\,x.\overline{c}.\overline{a}.x$. The opposite inclusion does not hold either. In fact, we have $a.b \preccurlyeq^{\mathsf{P}} a$, but $a.b \overset{\circ}{\not\preccurlyeq} a$, since, for the client $\overline{a}.\overline{b}$ we have that $\overline{a}.\overline{b} \dashv a.b$, but for no respectful f it is possible to have $f : \overline{a}.\overline{b} \dashv\!\vdash a$.

For what concerns the decidability of respectfulness, some other approaches could possibly be undertaken for its proof. For instance, one might define respectfulness as a μ-calculus formula. Decidability might then be obtained by model-checking the (finite) state space of the orchestrator against the formula.[5] The feasibility of such an approach with respect to the one of the present paper could be worth investigating.

An approach to the formal description of service contracts in terms of automata has been recently developed in [19]. The notion of *contract automaton* is related to that of *contract* as well as of *session contract*. Besides, the notion of *contract agreement* in [19] somewhat resembles that of *compliance*. In the framework of that paper, orchestrators are synthesised to enforce contract composition adhering to the requirements for contract agreement. Even if the authors of [19] work on the overall satisfaction in a multiparty composition of principals, it is definitely worthwhile, as a future investigation, to study the relation between the notion of orchestration, as developed in [11] and in the present paper, and the approach of [19], which in turn has been related in [20] to the model of choreography of communicating finite state machines (CFSM) [21]. For what concerns *session contracts* in particular, the investigation of the correspondence with the above mentioned formalisms could move from the result concerning the correspondence of *binary* session types with a particular two-communicating-machines subclass (see [22] for references). Such a correspondence between session types and communicating machines has been pushed further to the multiparty setting in [22].

Many properties of the model of CFSM which are intractable cease to be so when bags instead of – or together with – FIFO queues are taken into account [23]. The similarity of contracts and session contracts with the CFSM model suggests the investigation of the use of bags for session-contract interactions to reduce decidability problems in our context to problems in the CFSM model with bags. What does a bag correspond to in our context is however unclear. In fact, by putting a bag in between $\overline{a}.\overline{b}$ and $a + b$ would result in a number of possible non-deterministic evolutions of the system: as soon as a is in the bag, it could be used as input for the server; or, in case both a and b get into the bag, the server could non-deterministically choose amongst them; etc. Such a behaviour of the system, however, goes far beyond the session setting we are in, where non-determinism is restricted to occur only inside the client and server.

In [24] a two-players game-theoretic interpretation on event structures is provided for client–server systems of session contracts. Starting from an observation by Bartoletti, a three-players game interpretation of retractable contracts has been developed in [25], where the retractable actions correspond to moves of the third player, whose goal is having the first player win. Such a goal resembles that of an orchestrator. In fact it has been shown that the winning strategies for the third player are in one-to-one correspondence with compliance-enabling orchestrators for client–server systems of session contracts (where just particular input–output actions can actually be orchestrator-driven). Starting from [25] we aim at extending the correspondence orchestrator/strategy in the setting of multiparty sessions.

Decidability of $\dashv\!\vdash$ is an open problem worth investigating, in particular the possibility of finding a bound for the bounded version of the algorithm **Synth** as discussed in Remark 6.13.

In the context of Service Oriented Programming it is extremely valuable in practice to have a metric on the space of the possible servers for a given client. By means of such a metric the most suitable server could be selected from the set of the possible servers. For instance, in papers like [26,27], metrics are specified enabling to find the most suitable resources/services in either Grid or Cloud systems.

In a general setting as the one considered in the present paper, a possible metric on servers could be based on an abstract complexity measure of the orchestrators enabling the servers for a given client to be compliant with them. An

[5] As suggested by an anonymous referee.

obvious candidate for measuring orchestrators is the least size of buffers of the orchestrator. Then an interesting problem is to find orchestrators with minimal complexity.

Acknowledgements

This work was partially supported by COST Action IC1201 BETTY, Torino University/Compagnia San Paolo Project SALT and by the Project FIR 1B8C1 of the University of Catania. We are grateful to the referees for their helpful and meaningful advices. We also wish to thank Mariangiola Dezani for her everlasting support and Ivan Lanese for a useful remark.

Appendix A. Proofs of Section 3

We begin by showing that the disrespectful orchestrated compliance can be coinductively characterised. Such a characterisation will correspond to the (algorithmic) formal system of Definition 3.1.

A.1. Coinductive characterisation for disrespectful orchestrated compliance

We recall again that, when not specified otherwise, we take the equi-recursive view of recursion, by equating $\mathsf{rec}\, x.\sigma$ with $\sigma\{\mathsf{rec}\, x.\sigma/x\}$.

Definition A.1 *(Coinductive disrespectful orchestrated compliance).* Let $\{\dashv\!\vdash^{\mathsf{d}}_k \mid k \in \mathbb{N}\}$ be the family of relations over $\mathsf{Orch} \times \mathsf{SC} \times \mathsf{SC}$ such that

1. $\dashv\!\vdash^{\mathsf{d}}_0 = \mathsf{Orch} \times \mathsf{SC} \times \mathsf{SC}$ and
2. $f : \rho \ \dashv\!\vdash^{\mathsf{d}}_{k+1} \ \sigma$ if either:
   (a) $\rho = \mathbf{1}$ and $f = \mathbf{1}$; or
   (b) $\rho \neq \mathbf{1}$, $\rho \parallel_f \sigma \longrightarrow$, and $\rho \parallel_f \sigma \longrightarrow \rho' \parallel_{f'} \sigma'$ implies $f' : \rho' \ \dashv\!\vdash^{\mathsf{d}}_k \ \sigma'$, for all f', ρ', σ'.

Then we define $\dashv\!\vdash^{\mathsf{d}}_{co} = \bigcap_{k\in\mathbb{N}} \dashv\!\vdash^{\mathsf{d}}_k$.

Proposition A.2. *For any ρ, ρ, and σ,*

$$f : \rho \dashv\!\vdash^{\mathsf{d}}_{co} \sigma \quad \Leftrightarrow \quad f : \rho \dashv\!\vdash^{\mathsf{d}} \sigma .$$

Proof. The left-to-right implication is immediate by definition. Vice versa, by contraposition, let $f : \rho \not\dashv\!\vdash^{\mathsf{d}} \sigma$ and let k be the minimal natural number such that $f : \rho \not\dashv\!\vdash^{\mathsf{d}}_k \sigma$. Then there exists a sequence of reductions

$$\rho \parallel_f \sigma \longrightarrow \rho_1 \parallel_{f_1} \sigma_1 \longrightarrow \cdots \longrightarrow \rho_{k-1} \parallel_{f_{k-1}} \sigma_{k-1}$$

of length $k-1$ such that $\rho_i \parallel_{f_i} \sigma_i \longrightarrow$ for all $i < k-1$ and $f_{k-1} : \rho_{k-1} \not\dashv\!\vdash^{\mathsf{d}}_1 \sigma_{k-1}$. Therefore $\rho_{k-1} \neq \mathbf{1}$ or $f_{k-1} \neq \mathbf{1}$ and $\rho_{k-1} \parallel_{f_{k-1}} \sigma_{k-1} \not\longrightarrow$, which implies $\rho \not\dashv\!\vdash^{\mathsf{d}} \sigma$. □

We recall that we use 'coinductively compliant' as short for 'coinductively disrespectful orchestrated compliant'.
We formalise now the notion of semantics, with respect to which the system $\rhd$ will be proved sound and complete.

Definition A.3 *(A semantics for system $\rhd$).*

i) $\models \Gamma \triangleq \forall (f : \rho' \dashv^{\mathsf{d}} \sigma') \in \Gamma \ [\, f : \rho' \dashv\!\vdash^{\mathsf{d}} \sigma' \,]$;
ii) $\Gamma \models f : \rho \dashv^{\mathsf{d}} \sigma \triangleq \models \Gamma \implies f : \rho \dashv\!\vdash^{\mathsf{d}} \sigma$.

To facilitate the soundness and completeness proofs it is convenient to consider the following stratified version of Definition A.3:

Definition A.4 *(A stratified semantics for $\rhd$).*

i) $\models_k \Gamma \triangleq \forall (f : \rho' \dashv^{\mathsf{d}} \sigma') \in \Gamma \ [\, f : \rho' \dashv\!\vdash^{\mathsf{d}}_k \sigma' \,]$;
ii) $\Gamma \models_k f : \rho \dashv^{\mathsf{d}} \sigma \triangleq \models_k \Gamma \implies f : \rho \dashv\!\vdash^{\mathsf{d}}_k \sigma$.

From $\dashv\!\vdash^{\mathsf{d}}_{k+1} \subseteq \dashv\!\vdash^{\mathsf{d}}_k$ we have that $\models_{k+1} \Gamma$ implies $\models_k \Gamma$. Also, it is immediate to verify the following:

Fact A.5. $\forall k \, [\, \Gamma \models_k f : \rho \dashv^{\mathsf{d}} \sigma \,] \implies \Gamma \models f : \rho \dashv^{\mathsf{d}} \sigma$.

The opposite implication of Fact A.5 does not hold, as shown by the following example. Consider $\Gamma = \{\langle a, \overline{a}\rangle.1 : \overline{a}.c \dashv^{\mathsf{d}} a\}$ and $1 : b \dashv^{\mathsf{d}} \mathbf{1}$. Trivially, $\models 1 : b \dashv^{\mathsf{d}} \mathbf{1}$ does not hold. Moreover, it is easy to check that $\not\models \Gamma$. However, it is also possible to check that $\models_1 \Gamma$. In fact $\langle a, \overline{a}\rangle.1 : \overline{a}.c \dashv\!\mid^{\mathsf{d}}_1 a$ (since, trivially, $1 : c \dashv\!\mid^{\mathsf{d}}_0 \mathbf{1}$). So, $\Gamma \models 1 : b \dashv^{\mathsf{d}} \mathbf{1}$ holds simply because $\not\models \Gamma$, whereas we have that $\forall k\ [\,\Gamma \models_k 1 : b \dashv^{\mathsf{d}} \mathbf{1}\,]$ does not hold, since $\models_1 \Gamma$ but $\not\models_1 1 : b \dashv^{\mathsf{d}} \mathbf{1}$. As a matter of fact, the best we can say is that if $\Gamma \models \rho \dashv^{\mathsf{d}} \sigma$, then $\Gamma \models_k \rho \dashv^{\mathsf{d}} \sigma$ for all but finitely many k.

In order to prove the Completeness property, however, we will only need the following statement to hold: $\forall k\ [\models_k f : \rho \dashv^{\mathsf{d}} \sigma\,] \Leftrightarrow \models f : \rho \dashv^{\mathsf{d}} \sigma$.

Now we can show that the formal system is sound with respect to the judgement semantics. Notice that we are actually able to prove a stronger statement than that of Theorem 3.3.

Theorem A.6 *(Soundness). If* $\Gamma \rhd \rho \dashv^{\mathsf{d}} \sigma$, *then* $\Gamma \models \rho \dashv^{\mathsf{d}} \sigma$.

Proof. In view of Fact A.5 it suffices to prove that:

$$\Gamma \rhd f : \rho \dashv^{\mathsf{d}} \sigma \ \text{ implies } \ \Gamma \models_k f : \rho \dashv^{\mathsf{d}} \sigma \text{ for all } k.$$

We proceed by simultaneous induction over the derivation $\mathcal{D} :: \Gamma \rhd f : \rho \dashv^{\mathsf{d}} \sigma$ and over k. Since $\Gamma \models_0 f : \rho \dashv^{\mathsf{d}} \sigma$ trivially holds, we shall keep the case $k = 0$ implicit. We distinguish the possible cases of the last rule in $\mathcal{D}$.

Case (Ax): Then $\mathcal{D}$ consists of the inference

$$\frac{}{\Gamma \rhd 1 : \mathbf{1} \dashv^{\mathsf{d}} \sigma}\ (\text{Ax})$$

and the thesis is immediate since $1 : \mathbf{1} \dashv\!\mid^{\mathsf{d}}_k \sigma$ for all k;

Case (Hyp): Then $\mathcal{D}$ consists of the inference:

$$\frac{}{\Gamma, f : \rho \dashv^{\mathsf{d}} \sigma \rhd f : \rho \dashv^{\mathsf{d}} \sigma}\ (\text{Hyp})$$

and $\Gamma, f : \rho \dashv^{\mathsf{d}} \sigma \models_k f : \rho \dashv^{\mathsf{d}} \sigma$ holds trivially for all k.

Case (CplΣ-L): Then $\mathcal{D}$ has the form:

$$\frac{\Gamma' \rhd f' : \rho_p \dashv^{\mathsf{d}} \sigma \qquad (p \in I)}{\Gamma \rhd \langle \overline{a}_p, \varepsilon\rangle.f' : \Sigma_{i\in I} a_i.\rho_i \dashv^{\mathsf{d}} \sigma}\ (\text{CplΣ-L})$$

where $\Gamma' = \Gamma,\ \langle \overline{a}_p, \varepsilon\rangle.f' : \Sigma_{i\in I} a_i.\rho_i \dashv^{\mathsf{d}} \sigma$.

We have to prove that $\Gamma \models_k \langle \overline{a}_p, \varepsilon\rangle.f' : \Sigma_{i\in I} a_i.\rho_i \dashv^{\mathsf{d}} \sigma$ for all k.

Let $k > 0$; assume, by induction over k, $\Gamma \models_{k-1} \langle \overline{a}_p, \varepsilon\rangle.f' : \Sigma_{i\in I} a_i.\rho_i \dashv^{\mathsf{d}} \sigma$. If $\models_k \Gamma$, then $\models_{k-1} \Gamma$, which implies $\langle \overline{a}_p, \varepsilon\rangle.f' : \Sigma_{i\in I} a_i.\rho_i \dashv\!\mid^{\mathsf{d}}_{k-1} \sigma$ and hence $\models_{k-1} \Gamma'$, since $\Gamma' = \Gamma, \langle \overline{a}_p, \varepsilon\rangle.f' : \Sigma_{i\in I} a_i.\rho_i \dashv^{\mathsf{d}} \sigma$. By induction over $\mathcal{D}$ we know that $\Gamma' \models_h f' : \rho_p \dashv^{\mathsf{d}} \sigma$ for $p \in I$ and for all h, hence $\Gamma' \models_{k-1} f' : \rho_p \dashv^{\mathsf{d}} \sigma$.

Combining this with $\models_{k-1} \Gamma'$ we get $f' : \rho_p \dashv\!\mid^{\mathsf{d}}_{k-1} \sigma$, the only one-step reduct of $\Sigma_{i\in I} a_i.\rho_i \parallel_f \sigma$, where $f = \langle \overline{a}_p, \varepsilon\rangle.f'$, is $\rho_p \parallel_f \sigma$, so we conclude $\Sigma_{i\in I} a_i.\rho_i \dashv\!\mid^{\mathsf{d}}_k \sigma$ as desired.

Cases (CplΣ-R), (Cpl⊕-R) and (Cpl⊕-L): These cases can be treated similarly to the previous one.

Case (Cpl⊕-Σ): Then $\mathcal{D}$ ends by:

$$\frac{\Gamma' \rhd f_i : \rho_i \dashv^{\mathsf{d}} \Sigma_{j\in J} a_j.\sigma_j\ (\forall i \in H) \quad \Gamma' \rhd f_i : \rho_i \dashv^{\mathsf{d}} \sigma_i\ (\forall i \in K)\ (K \subseteq J)}{\Gamma \rhd f : \oplus_{i\in H\cup K} \overline{a}_i.\rho_i \dashv^{\mathsf{d}} \Sigma_{j\in J} a_j.\sigma_j}\ (\text{Cpl}\oplus\text{-}\Sigma)$$

where $\Gamma' = \Gamma,\ f : \oplus_{i\in H\cup K} \overline{a}_i.\rho_i \dashv^{\mathsf{d}} \Sigma_{j\in J} a_j.\sigma_j$ and $f = (\bigvee_{h\in H} \langle a_h, \varepsilon\rangle.f_h) \vee (\bigvee_{k\in K} \langle a_k, \overline{a}_k\rangle.f_k)$.

We have to prove that $\Gamma \models_k \langle \overline{a}_p, \varepsilon\rangle.f' : \Sigma_{i\in I} a_i.\rho_i \dashv^{\mathsf{d}} \sigma$ for all k.

Let $k > 0$; assume, by induction over k,

$$\Gamma \models_{k-1} \bigvee_{h\in H} \langle a_h, \varepsilon\rangle.f_h) \vee (\bigvee_{k\in K} \langle a_k, \overline{a}_k\rangle.f_k : \bigoplus_{i\in I} \overline{a}_i.\rho_i \dashv^{\mathsf{d}} \sum_{j\in J} a_j.\sigma_j$$

If $\models_k \Gamma$, then $\models_{k-1} \Gamma$, which implies $\bigvee_{h\in H} \langle a_h, \varepsilon\rangle.f_h) \vee (\bigvee_{k\in K} \langle a_k, \overline{a}_k\rangle.f_k : \oplus_{i\in I} \overline{a}_i.\rho_i \dashv\!\mid^{\mathsf{d}}_{k-1} \Sigma_{j\in J} a_j.\sigma_j$ and hence $\models_{k-1} \Gamma'$, since

$$\Gamma' = \Gamma,\ \bigvee_{h\in H} \langle a_h, \varepsilon\rangle.f_h) \vee (\bigvee_{k\in K} \langle a_k, \overline{a}_k\rangle.f_k : \bigoplus_{i\in I} \overline{a}_i.\rho_i \dashv^{\mathsf{d}} \Sigma_{j\in J} a_j.\sigma_j.$$

By induction over $\mathcal{D}$ we know that for all h, and hence in particular for $k-1$, $\Gamma' \models_{k-1} f_i : \rho_i \dashv^{\mathsf{d}} \Sigma_{j\in J} a_j.\sigma_j$ for all $i \in H$ and $\Gamma' \models_{k-1} f_i : \rho_i \dashv^{\mathsf{d}} \sigma_i$ for all $i \in K$, where $K \subseteq J$. Combining this with $\models_{k-1} \Gamma'$ we get $f_i : \rho_i \dashv\!\mid^{\mathsf{d}}_{k-1} \Sigma_{j\in J} a_j.\sigma_j$

for all $i \in H$ and $f_i : \rho_i \dashv\!\!\mid^{\mathsf{d}}_{k-1} \sigma_i$ for all $i \in K$, where $K \subseteq J$, $f' : \rho_p \dashv\!\!\mid^{\mathsf{d}}_{k-1} \sigma$. Now, since the one-step reducts of $\oplus_{i\in H\cup K} \overline{a}_i.\rho_i \parallel_f \Sigma_{j\in J} a_j.\sigma_j$, where $f = \bigvee_{h\in H}\langle a_h, \varepsilon\rangle.f_h) \vee (\bigvee_{k\in K}\langle a_k, \overline{a}_k\rangle.f_k$ are exactly $\rho_i \parallel_{f_i} \Sigma_{j\in J} a_j.\sigma_j$ for all $i \in H$ and $\rho_i \parallel_{f_i} \sigma_i$ for all $i \in K$, where $K \subseteq J$, we conclude $\Sigma_{i\in I} a_i.\rho_i \dashv\!\!\mid^{\mathsf{d}}_k \sigma$ as desired.

Case (CPLΣ-$\oplus$): This case can be treated similarly to the previous one. □

We can now prove Lemma 3.5, assuring that a failure of the algorithm **Prove** can only happen if the configurations are not compliant.

Lemma A.7 (***Lemma 3.5***). *If* **Prove** $(\Gamma \rhd f : \rho \dashv^{\mathsf{d}} \sigma) =$ **fail**, *then* $f : \rho \not\dashv\!\!\mid^{\mathsf{d}} \sigma$.

Proof. Observe that if **Prove** $(\Gamma \rhd f : \rho \dashv^{\mathsf{d}} \sigma) =$ **fail**, then $f : \rho \dashv^{\mathsf{d}} \sigma \notin \Gamma$. This will be tacitly assumed in all cases below.

Let k be the maximum number of recursive calls of the terminating execution of

$$\textbf{Prove}\,(\Gamma \rhd f : \rho \dashv^{\mathsf{d}} \sigma)$$

returning **fail**. We prove by induction over k that there exists h (actually greater than k) such that $f : \rho \not\dashv\!\!\mid^{\mathsf{d}}_h \sigma$. This suffices since $\dashv\!\!\mid^{\mathsf{d}}_{co} = \bigcap_k \dashv\!\!\mid^{\mathsf{d}}_k$ by Lemma A.2.

$k = 0$: Then **Prove** $(\Gamma \rhd f : \rho \dashv^{\mathsf{d}} \sigma) =$ **fail** implies $\rho \neq \mathbf{1}$ or $f \neq 1$ (otherwise **Prove** succeeds). In case $f = 1$ we get trivially that $\rho \parallel_f \sigma \not\rightarrow$ and hence $f : \rho \not\dashv\!\!\mid^{\mathsf{d}}_h \sigma$ for any $h > 0$. Otherwise, if $f \neq 1$ we have to take into account the other cases for which **Prove** $(\Gamma \rhd f : \mathbf{1} \dashv^{\mathsf{d}} \sigma) =$ **fail**.
Take the case $f = \langle \varepsilon, \overline{a}_p\rangle.f'$ and $\sigma = \oplus_{i\in I} a_i.\sigma_i$ (all other cases can be treated similarly). Then the only possible reductions, for any $i \in I$, are

$$\rho \parallel_f \oplus_{i\in I} a_i.\sigma_i \rightarrow \rho \parallel_f a_i.\sigma_i \not\rightarrow$$

so forcing $f : \rho \not\dashv\!\!\mid^{\mathsf{d}}_h \sigma$ for any $h > 1$.

$k > 0$: Necessarily $\rho \neq \mathbf{1}$ or $f \neq 1$. The negative result of the computation depends on the failure of some recursive call. Since all cases are similar, we consider for example the case when $f = \bigvee_{j\in J}\langle \varepsilon, b_j\rangle.f_j$ and $\sigma = \oplus_{j\in J}\overline{b}_j.\sigma_i$. Then **Prove** $(\Gamma' \rhd f_j : \rho \dashv^{\mathsf{d}} \sigma_j) =$ **fail** for at least one $j \in J$, say q. By induction we get that $f_q : \rho \not\dashv\!\!\mid^{\mathsf{d}}_h \sigma_q$. Since $\rho \parallel_{f_q} \sigma_q$ is a reduct of $\rho \parallel_f \oplus_{j\in J}\overline{b}_j.\sigma_i$, we get $f : \rho \not\dashv\!\!\mid^{\mathsf{d}}{}_{h+1} \oplus_{j\in J}\overline{b}_j.\sigma_i$. □

We proceed now towards the proof that **Prove** always terminates. We begin by defining the set of subexpressions of a behaviour and of an orchestrator as expected. We overload the definition of the function Sub below for the sake of readability.

Definition A.8 *(Subexpressions).*

i) The function $\mathsf{Sub} : \mathsf{SC} \to \wp(\mathsf{SC})$ is coinductively given by:

$$\begin{aligned}
\mathsf{Sub}(\mathbf{1}) &= \{\mathbf{1}\}\\
\mathsf{Sub}(\Sigma_{i\in I} a_i.\sigma_i) &= \{\Sigma_{i\in I} a_i.\sigma_i\} \cup \textstyle\bigcup_{i\in I} \mathsf{Sub}(\sigma_i)\\
\mathsf{Sub}(\oplus_{i\in I} \overline{a}_i.\sigma_i) &= \{\oplus_{i\in I} \overline{a}_i.\sigma_i\} \cup \textstyle\bigcup_{i\in I} \mathsf{Sub}(\sigma_i).
\end{aligned}$$

ii) The function $\mathsf{Sub} : \mathsf{Orch} \to \wp(\mathsf{Orch})$ is coinductively given by:

$$\begin{aligned}
\mathsf{Sub}(1) &= \{\mathbf{1}\}\\
\mathsf{Sub}(\mu.f) &= \{\mu.f\} \cup \mathsf{Sub}(f)\\
\mathsf{Sub}(\textstyle\bigvee_{i\in I} f_i) &= \{\textstyle\bigvee_{i\in I} f_i\} \cup \textstyle\bigcup_{i\in I} \mathsf{Sub}(\iota_{L\,i}.f_i).
\end{aligned}$$

Since we have assumed that $\mathsf{rec}\,x.\sigma = \sigma\{\mathsf{rec}\,x.\sigma/x\}$ and $\mathsf{rec}\,x.f = f\{\mathsf{rec}\,x.f/x\}$, behaviours and orchestrators containing (proper) recursive subterms are infinite terms, hence the coinductive character of Sub; in particular we have that $\mathsf{Sub}(\mathsf{rec}\,x.\sigma) = \mathsf{Sub}(\sigma\{\mathsf{rec}\,x.\sigma/x\})$ and $\mathsf{Sub}(\mathsf{rec}\,x.f) = \mathsf{Sub}(f\{\mathsf{rec}\,x.f/x\})$.

On the other hand, recursion being guarded, σ is a regular tree. A similar argument holds for orchestrators. Hence:

Fact A.9.

i) *For any σ, the set* $\mathsf{Sub}(\sigma)$ *is well defined and finite.*
ii) *For any f, the set* $\mathsf{Sub}(f)$ *is well defined and finite.*

We can now provide the proof of Lemma 3.6.

$$\overline{\mathsf{cTs}}^a(1) = 1 \qquad \overline{\mathsf{cTs}}^a(x) = x$$

$$\overline{\mathsf{cTs}}^a(\langle \varepsilon, \overline{a}\rangle.f') = \overset{\circ}{\underset{\overline{\mathsf{cTs}}^a(f')}{0 \mid -1}} \qquad \overline{\mathsf{cTs}}^a(\langle a, \varepsilon\rangle.f') = \overset{\circ}{\underset{\overline{\mathsf{cTs}}^a(f')}{+1 \mid 0}}$$

$$\overline{\mathsf{cTs}}^a(\langle \overline{a}, \varepsilon\rangle.f') = \overset{\circ}{\underset{\overline{\mathsf{cTs}}^a(f')}{-1 \mid 0}} \qquad \overline{\mathsf{cTs}}^a(\langle \varepsilon, a\rangle.f') = \overset{\circ}{\underset{\overline{\mathsf{cTs}}^a(f')}{0 \mid +1}}$$

$$\overline{\mathsf{cTs}}^a(\mu.f') = \overset{\circ}{\underset{\overline{\mathsf{cTs}}^a(f')}{0 \mid 0}} \text{ if } \mu \notin \{\langle \varepsilon, \overline{a}\rangle, \langle a, \varepsilon\rangle, \langle \overline{a}, \varepsilon\rangle, \langle \varepsilon, a\rangle\}$$

$$\overline{\mathsf{cTs}}^a(f_1 \vee \ldots \vee f_n) = \overset{\circ}{\underset{\overline{\mathsf{cTs}}^a(f_1)\ldots\overline{\mathsf{cTs}}^a(f_n)}{0 \diagup 0 \; 0 \diagdown 0}} \qquad \overline{\mathsf{cTs}}^a(\mathsf{rec}\, x.f') = \overset{\mathsf{rec}\, x}{\underset{\overline{\mathsf{cTs}}^a(f')}{0 \mid 0}}$$

Fig. B.6. Buffer-aware a-tree.

Lemma A.10 (*Lemma 3.6*). *For all judgements* $\Gamma \rhd f : \rho \dashv^{\mathsf{d}} \sigma$ *the execution of* **Prove** $(\Gamma \rhd f : \rho \dashv^{\mathsf{d}} \sigma)$ *terminates.*

Proof. Given a statement $f : \rho \dashv^{\mathsf{d}} \sigma$ we set:

$$\mathsf{Sub}(f : \rho \dashv^{\mathsf{d}} \sigma) = \{ f' : \rho' \dashv^{\mathsf{d}} \sigma' \mid f' \in \mathsf{Sub}(f), \rho' \in \mathsf{Sub}(\rho), \sigma' \in \mathsf{Sub}(\sigma) \}.$$

Fact A.9 implies that $\mathsf{Sub}(f : \rho \dashv^{\mathsf{d}} \sigma)$ is finite. On the other hand, by direct inspection of the rules of the system in Fig. 3, we find that all $f' : \rho' \dashv^{\mathsf{d}} \sigma'$ occurring in the premises belong to the set $\mathsf{Sub}(f : \rho \dashv^{\mathsf{d}} \sigma)$ for some $f : \rho \dashv^{\mathsf{d}} \sigma$ that occurs in the conclusion.

Now, if **Prove** $(\Gamma \rhd f : \rho \dashv^{\mathsf{d}} \sigma)$ does not terminate, then there exists an infinite sequence of nested calls **Prove** $(\Gamma_0 \rhd f_0 : \rho_0 \dashv^{\mathsf{d}} \sigma_0)$, **Prove** $(\Gamma_1 \rhd f_1 : \rho_1 \dashv^{\mathsf{d}} \sigma_1), \ldots$, where $\Gamma_0 \rhd f_0 : \rho_0 \dashv^{\mathsf{d}} \sigma_0$ is just $\Gamma \rhd f : \rho \dashv^{\mathsf{d}} \sigma$, and the sequence $\Gamma_0, \Gamma_1, \ldots$ is such that $\Gamma_{i+1} = \Gamma_i \cup \{ f_i : \rho_i \dashv^{\mathsf{d}} \sigma_i \}$ for all i. Since **Prove** begins by checking $f : \rho \dashv^{\mathsf{d}} \sigma \in \Gamma$ and returns in the positive case, non-termination would only be possible if $\Gamma_i \subset \Gamma_{i+1}$ for infinitely many i, contradicting the fact that each Γ_i is a subset of the union of Γ and $\mathsf{Sub}(f : \rho \dashv^{\mathsf{d}} \sigma)$, which are both finite sets. □

Appendix B. Proof of respectfulness decidability (Theorem 4.7)

In order to show decidability of the respectfulness property, we provide a characterisation of respectfulness based on the notion of buffer-aware trees and its related labellings below.

In the present subsection we treat orchestrators as syntactical expressions so that, contrary to the convention we have so far been following, we distinguish $\mathsf{rec}\, x.f$ from $f\{\mathsf{rec}\, x.f/x\}$, although they denote the same orchestrating process.

Definition B.1 (*Buffer-aware trees of* f).

1. Let $a \in \mathcal{N}$. We define the *buffer-aware* a*-tree of an orchestrator* f, denoted by $\overline{\mathsf{cTs}}^a(f)$, as the tree defined by induction over the expression f in Fig. B.6. The edges of the tree have a left and a right-weight denoting, respectively, the increment of the client-to-server and of the server-to-client buffer for the name a caused by the orchestration action, if any, performed by f.
2. Given an edge e of a buffer-aware a-tree t, we denote its left (resp. right) weight by $\mathsf{lw}^t(e)$ (resp. $\mathsf{rw}^t(e)$).
3. We define the *buffer-aware* $*$*-tree of an orchestrator* f, denoted by $\overline{\mathsf{cTs}}^*(f)$, as the tree with the same nodes and edges as $\overline{\mathsf{cTs}}^a(f)$, but such that the left (resp. right) weight of an edge e is $\Sigma_{a \in \mathcal{N}} \mathsf{lw}^{\overline{\mathsf{cTs}}^a(f)}(e)$ (resp. $\Sigma_{a \in \mathcal{N}} \mathsf{rw}^{\overline{\mathsf{cTs}}^a(f)}(e)$).

Notice that all but just one of them the terms in the summands in Definition B.1(3) above are 0.

Fact B.2. *The left and right weights of the edges of a buffer-aware* $*$*-tree of an orchestrator* f *are either* 0, -1, *or* $+1$.

Definition B.3 (*Buffer-labelling of* $\overline{\mathsf{cTs}}^a(f)$). We define the *buffer-labelling of* $\overline{\mathsf{cTs}}^a(f)$ by labelling its nodes with left and right-labels as follows: given a node N and the path P in $\overline{\mathsf{cTs}}^a(f)$ from the root to N, we left-label N with the sum of all the left-weights of the edges in P, whereas we right-label N with the sum of all the right-weights of the edges in P.

We denote with $\mathsf{ll}^{\overline{\mathsf{cTs}}^a(f)}(N)$ (resp. $\mathsf{rl}^{\overline{\mathsf{cTs}}^a(f)}(N)$) the left (resp. right) label of the node N in the buffer-labelling of $\overline{\mathsf{cTs}}^a(f)$.

We now provide characterisations for the properties defining respectfulness.

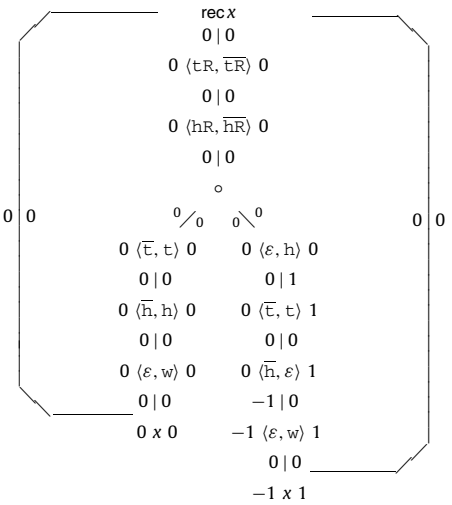

Fig. B.7. The graph $\mathsf{cGs}^{\mathrm{h}}(\mathsf{f})$ for the orchestrator f of the Introduction.

Definition B.4 *(Sound buffer-labelling).* The buffer-labelling of $\mathsf{c\overline{Ts}}^{a}(f)$ is *sound* whenever:

1. there is no node with either a negative left-label or a negative right-label and
2. for any leaf x and corresponding $\mathsf{rec}\,x$ node, if k is the left (resp. right) label of x and h is the left (resp. right) label of $\mathsf{rec}\,x$, then: $k-h \geq 0$.

We will now show that f is sound if and only if the buffer-labelling of $\mathsf{c\overline{Ts}}^{a}(f)$ is, for all names a in f.

Definition B.5 *(Buffer-aware graphs of f).*

1. Given an a-tree of an orchestrator f, the *corresponding* $\mathsf{rec}\,x$ node of a leaf x is the node corresponding to the binder $\mathsf{rec}\,x$ in f which binds the variable x in f.
2. The *a-graph of an orchestrator f*, (resp. *$*$-graph of an orchestrator f*) denoted by $\mathsf{cGs}^{a}(f)$ (resp.$\mathsf{cGs}^{*}(f)$) is the graph obtained out of $\mathsf{c\overline{Ts}}^{a}(f)$ (resp. $\mathsf{c\overline{Ts}}^{*}(f)$) by connecting any leaf x with its corresponding $\mathsf{rec}\,x$ node. The new edge is right- and left-labelled by 0.
   We call the node corresponding to the root of the tree out of which the graph has been built the *root of the graph*. We call the set of edges in a graph connecting a $\mathsf{rec}\,x$ node to itself a *cycle*.

Example B.6. Consider the following orchestrator, as defined in the Introduction

$$\begin{aligned}\mathsf{f} \;=\; \mathsf{rec}\,x.\langle \mathtt{tR}, \overline{\mathtt{tR}}\rangle.\langle \mathtt{hR}, \overline{\mathtt{hR}}\rangle.(&\langle \overline{\mathtt{t}}, \mathtt{t}\rangle.\langle \overline{\mathtt{h}}, \mathtt{h}\rangle.\langle \varepsilon, \mathtt{w}\rangle.x\\ &\vee\\ &\langle \varepsilon, \mathtt{h}\rangle.\langle \overline{\mathtt{t}}, \mathtt{t}\rangle.\langle \overline{\mathtt{h}}, \varepsilon\rangle.\langle \varepsilon, \mathtt{w}\rangle.x)\end{aligned}$$

According to the above definitions, the graph in Fig. B.7 is $\mathsf{cGs}^{\mathrm{h}}(\mathsf{f})$.

Since $\mathsf{rec}\,x.f$ with $x \notin \mathrm{FN}(f)$ is semantically the same as f so that it is harmless to delete $\mathsf{rec}\,x$ in such a case, we shall assume below (without loss of generality) that all bound variables actually occur in the scope of their respective binders.

Fact B.7.

1. *Each $\boldsymbol{\mu} \in \mathsf{MaxTr}(f)$ corresponds to a path of maximal length in $\mathsf{cGs}^{a}(f)$ starting from the root, and vice versa.*
2. *Let $\boldsymbol{\mu} \in \mathsf{Tr}(f)$ and let E be the multiset of the edges of $\mathsf{cGs}^{a}(f)$ in the path corresponding to $\boldsymbol{\mu}$. Then*

$$\begin{aligned}{}_{a}|\tilde{\emptyset}\mu_1 \cdots \mu_n| \;&=\; \Sigma_{e\in E}\, \mathsf{lw}^{\mathsf{c\overline{Ts}}^{a}(f)}(e) \;\; \textit{and}\\ |\tilde{\emptyset}\mu_1 \cdots \mu_n|_{a} \;&=\; \Sigma_{e\in E}\, \mathsf{rw}^{\mathsf{c\overline{Ts}}^{a}(f)}(e)\end{aligned}$$

Proposition B.8. *f is sound if and only if the buffer-labelling of $\mathsf{cTs}^a(f)$ is sound, for any $a \in \mathcal{N}$.*

Proof.

$\Leftarrow$: Let $\boldsymbol{\mu} \in \mathsf{MaxTr}(f)$ be arbitrary, a a name used in f and take $n \leq |\boldsymbol{\mu}|$. Observe that if C is the set of the edges of a cycle in $\mathsf{cGs}^a(f)$, then condition B.4(2) ensures that

$$\Sigma_{e\in C}\mathsf{lw}^{\mathsf{cTs}^a(f)}(e) \geq 0 \quad \text{and} \quad \Sigma_{e\in C}\mathsf{rw}^{\mathsf{cTs}^a(f)}(e) \geq 0. \tag{B.1}$$

Consider now the path in $\mathsf{cGs}^a(f)$ corresponding to the subsequence $\mu_1 \cdots \mu_n$, and let E be the multiset of the edges of it. Condition B.4(1) together with property (B.1) above ensure that

$$\Sigma_{e\in E}\mathsf{lw}^{\mathsf{cTs}^a(f)}(e) \geq 0 \quad \text{and} \quad \Sigma_{e\in E}\mathsf{rw}^{\mathsf{cTs}^a(f)}(e) \geq 0.$$

The thesis then follows by Fact B.7(2)

$\Rightarrow$: By contraposition; assume that for a name $b \in \mathcal{N}$, the buffer-labelling of $\mathsf{cTs}^b(f)$ is unsound. Then we have two cases to consider:

(a) There is a negative label. Then any maximal trace having as prefix the trace corresponding to the path from the root to the node with negative label is unsound.

(b) There exists a leaf x and its corresponding $\mathsf{rec}\,x$ node, where k is the left (or right) label of x and h is the left (or right) label of $\mathsf{rec}\,x$, such that $k-h < 0$. Then we construct an unsound trace as follows. Take the path in $\mathsf{cGs}^b(f)$ starting from the root and cycling p times on the cycle, passing through the leaf x and its corresponding $\mathsf{rec}\,x$ node and ending on x. Let E be the multiset of edges of that path. Assume that k and h such that $k-h < 0$ are left-labels (the case of right-labels is treated similarly). Call N the leaf x we are considering. By definition we have that:

$$\Sigma_{e\in E}\mathsf{lw}^{\mathsf{cTs}^a(f)}(e) \;=\; \mathsf{ll}^{\mathsf{cTs}^a(f)}(N) + (p * (k-h))$$

Since $k-h < 0$, for a sufficiently large p we have that

$$\mathsf{ll}^{\mathsf{cTs}^a(f)}(N) + (p * (k-h)) < 0.$$

Therefore by Fact B.7(2) we get ${}_a|\tilde{\emptyset}\boldsymbol{\mu}| < 0$, where $\boldsymbol{\mu} \in \mathsf{Tr}(f)$ is the sequence corresponding to the path with p cycles. Therefore any $\boldsymbol{\mu'} \in \mathsf{MaxTr}(f)$ having $\boldsymbol{\mu}$ as prefix is unsound. □

We say that a node N in $\mathsf{cTs}^a(f)$ *gets to* 1 whenever there is a path in $\mathsf{cGs}^a(f)$ from N to a 1 node.

Definition B.9 *(Client-respectful buffer-labelling).* The buffer-labelling of $\mathsf{cTs}^a(f)$ is *client-respectful* whenever

1. any 1 node is left-labelled with 0;
2. for any leaf x and its corresponding $\mathsf{rec}\,x$ node, if k is the left-label of x and h is the left-label of $\mathsf{rec}\,x$, then
   (a) if the $\mathsf{rec}\,x$ node gets to 1, then $h = k$;
   (b) otherwise, if all the left-labels of the edges from $\mathsf{rec}\,x$ to x are 0 then $h = 0$;
3. for any path from a leaf x to its corresponding $\mathsf{rec}\,x$ node, either no edge is left-weighted with $+1$ or there is at least an edge with left-weight -1.

Proposition B.10. *f is client-respectful if and only if the buffer-labelling of $\mathsf{cTs}^a(f)$ is client-respectful, for any $a \in \mathcal{N}$.*

Proof.

$\Leftarrow$: Let $\boldsymbol{\mu} \in \mathsf{MaxTr}(f)$ be such that, for a given a used in f, ${}_a\!\downarrow\boldsymbol{\mu}$ is finite. Then ${}_a|\tilde{\emptyset}\boldsymbol{\mu}| = 0$ since, by Fact B.7(2), ${}_a|\tilde{\emptyset}\boldsymbol{\mu}| = \Sigma_{e\in E}\mathsf{lw}^{\mathsf{cTs}^a(f)}(e)$, where E is the multiset of edges of the path in $\mathsf{cGs}^a(f)$ corresponding to $\boldsymbol{\mu}$ (which ends in a 1 node). We have that $\Sigma_{e\in E}\mathsf{lw}^{\mathsf{cTs}^a(f)}(e) = 0$ by the conditions (1), (2a) and (2b) imposed on any client-respectful buffer labelling. Now consider the case when $\boldsymbol{\mu} \in \mathsf{MaxTr}(f)$ is such that ${}_a\!\downarrow\boldsymbol{\mu}$ is infinite, and take the path in $\mathsf{cGs}^a(f)$ corresponding to $\boldsymbol{\mu}$. If ${}_a\!\downarrow\boldsymbol{\mu}$ were definitely-$\langle a, \varepsilon\rangle$, all the edges in the path from a certain point on should have left-label $+1$. But this is impossible because of condition (3) of client-respectful buffer-labelling.

$\Rightarrow$: By contraposition; assume that for a name $b \in \mathcal{N}$, the buffer-labelling of $\mathsf{cTs}^b(f)$ is non-client-respectful. We consider the four possible cases:

(a) A label of a 1 leaf is not 0. In that case we immediately get a finite sequence out of f which is non-client-respectful, which is the sequence corresponding to the path in $\mathsf{cTs}^b(f)$ from the root to the 1 leaf.

(b) There is a node x labelled with k and its corresponding node $\mathsf{rec}\,x$ gets to 1 and it is labelled with h, with $k \neq h$. Consider the path in $\mathsf{cGs}^b(f)$ starting form the root and terminating in the 1 leaf after cycling $p > 0$ times in the x-$\mathsf{rec}\,x$ cycle. The sequence $\boldsymbol{\mu} \in \mathsf{MaxTr}(f)$ corresponding to this path is not client-respectful, since trivially ${}_b\!\downarrow\boldsymbol{\mu}$ is finite and ${}_b|\tilde{\emptyset}\boldsymbol{\mu}| = (p * (h-k)) \neq 0$.

(c) There is a node x labelled with k such that its corresponding $\mathsf{rec}\,x$ node, labelled with h, does not get to $\mathbf{1}$ and all the left-labels of the edges from $\mathsf{rec}\,x$ to x are 0. Besides, $h \neq 0$.
Consider the infinite path in $\mathsf{cGs}^b(f)$ starting form the root and keeping indefinitely cycling on the x-$\mathsf{rec}\,x$ cycle. The sequence $\boldsymbol{\mu} \in \mathsf{MaxTr}(f)$ corresponding to this path is not client-respectful, since ${}_b\!\downarrow\boldsymbol{\mu}$ is finite and $|\tilde{\mathcal{O}}({}_b\!\downarrow\boldsymbol{\mu})| = h \neq 0$.

(d) There exists a path from a leaf x to its corresponding $\mathsf{rec}\,x$ such that there are some edges left-weighted with $+1$ and no edge with left-weight -1.
Consider the infinite path in $\mathsf{cGs}^b(f)$ starting form the root and keeping indefinitely cycling on the x-$\mathsf{rec}\,x$ cycle. The sequence $\boldsymbol{\mu} \in \mathsf{MaxTr}(f)$ corresponding to this path is not client-respectful, since ${}_b\!\downarrow\boldsymbol{\mu}$ is infinite (because of the presence of the left-label $+1$ in the cycle) and from the point corresponding to the beginning of the infinite cycling the sequence is definitely-$\langle b, \varepsilon\rangle$ (because no -1 left-label is present in the cycle) and hence non-client-respectful. □

Definition B.11 *(Non-definitely server-inputted $*$-tree).* Given an orchestrator f, its $*$-tree $\mathsf{c\overline{Ts}}^*(f)$ is *non-definitely server-inputted* whenever, for any path from a leaf x to its corresponding $\mathsf{rec}\,x$ node, there is at least one edge with right-weight -1 or 0.

Proposition B.12. *f is non-definitely server-inputted, if and only if $\mathsf{c\overline{Ts}}^*(f)$ is non-definitely server-inputted.*

Proof.

⇐: Any infinite $\boldsymbol{\mu} \in \mathsf{MaxTr}(f)$ corresponds to an infinite path in $\mathsf{cGs}^*(f)$ starting from the root and consists of, from a certain point on, an infinite number of cycles. Since in each cycle there is at least an edge which is right-weighted with -1 or 0, it is impossible for $\boldsymbol{\mu}$ to be definitely server-inputted, otherwise at least one cycle should have all the edges with right-weight $+1$.

⇒: By contraposition, assume that $\mathsf{c\overline{Ts}}^*(f)$ is definitely server-inputted. This means that there is a path in $\mathsf{c\overline{Ts}}^*(f)$ from a leaf x to its corresponding $\mathsf{rec}\,x$ node with all the edges right-weighted with $+1$. We can therefore find an infinite $\boldsymbol{\mu} \in \mathsf{MaxTr}(f)$ which is definitely-$\{\langle \varepsilon, a\rangle \mid a \in \mathcal{N}\}$. It is enough to take the $\boldsymbol{\mu}$ corresponding to the infinite path in $\mathsf{c\overline{Ts}}^*(f)$ which starts from the root and cycles indefinitely of the x-$\mathsf{rec}\,x$ cycle. In fact, since all the right-weights in the cycle are $+1$, the sequence $\boldsymbol{\mu}$ is definitely-$\{\langle \varepsilon, a\rangle \mid a \in \mathcal{N}\}$. □

Lemma B.13. *For any orchestrator f and any $a \in \mathcal{N}$, the following properties of $\mathsf{c\overline{Ts}}^a(f)$ and $\mathsf{c\overline{Ts}}^*(f)$ are decidable:*

1. *The buffer-labelling of $\mathsf{c\overline{Ts}}^a(f)$ is* sound.
2. *The buffer-labelling of $\mathsf{c\overline{Ts}}^a(f)$ is* client-respectful.
3. *$\mathsf{c\overline{Ts}}^*(f)$ is non-definitely server-inputted.*

Proof. Immediate consequence of the fact that conditions in Definition B.4 (sound buffer labelling), B.9 (client-respectful labelling) and B.11 (non-definitely server-inputted) are mere checks of the labellings of edges in finite graphs. □

We can now show decidability of the respectfulness property for orchestrators as an immediate corollary of the previous propositions and lemmas.

Corollary B.14 *(Orchestrator-respectfulness decidability (**Theorem 4.7**)). Orchestrator respectfulness is decidable.*

Proof. Immediate from Proposition B.8, B.10, B.12 and Lemma B.13. □

Appendix C. System $\triangleright^{\mathsf{inf}}$

In this section we define an inference system $\triangleright^{\mathsf{inf}}$ for (possibly open) orchestrators, deducing judgements like $f : \rho \dashv^{\mathsf{d}} \sigma$, under finitely many assumptions of a certain shape. We will establish that the system is equivalent to System $\triangleright$ (i.e. they derive the same judgements) if we consider derivation with conclusions with empty environments end proper (i.e. closed) orchestrators. It is used in the proofs of Appendix D and is also at the basis of the algorithm **SB** for synthesising skipping orchestrators.

Definition C.1 *(The orchestrators inference system $\triangleright^{\mathsf{inf}}$).* The judgements of the system are expressions of the form $\Gamma \triangleright^{\mathsf{inf}} f : \rho \dashv^{\mathsf{d}} \sigma$, where $\rho, \sigma \in \mathsf{SC}$, f is a (possibly open) orchestrator and Γ is a set of assumptions of the form $x : \rho_i \dashv^{\mathsf{d}} \sigma_i$.
The axioms and rules of the system are described in Fig. C.8.

In the inference system of Fig. C.8, the symbol $\dashv^{\mathsf{d}}$ is a relation symbol representing the relation $\dashv\!\vdash^{\mathsf{d}}$ as defined in Definition 2.11. In order to give the intuition behind the inference system, let us briefly comment on one of the rules, say

$$(\text{Ax}):\ \frac{}{\Gamma \rhd^{\inf} 1 : \mathbf{1} \dashv^{\mathsf{d}} \sigma} \qquad (\text{Hyp}):\ \frac{}{\Gamma, x{:}\rho \dashv^{\mathsf{d}} \sigma \rhd^{\inf} x : \rho \dashv^{\mathsf{d}} \sigma}$$

$$(\text{Cpl}\Sigma\text{-L}):\ \frac{\Gamma, x{:}\,\Sigma_{i\in I}\, a_i.\rho_i \dashv^{\mathsf{d}} \sigma \rhd^{\inf} f' : \rho_p \dashv^{\mathsf{d}} \sigma}{\Gamma \rhd^{\inf} \mathsf{rec}\, x.\langle \overline{a}_p, \varepsilon\rangle.f' : \Sigma_{i\in I}\, a_i.\rho_i \dashv^{\mathsf{d}} \sigma}\ (p \in I)$$

$$(\text{Cpl}\Sigma\text{-R}):\ \frac{\Gamma, x{:}\rho \dashv^{\mathsf{d}} \Sigma_{i\in I}\, a_i.\sigma_i \rhd^{\inf} f' : \rho \dashv^{\mathsf{d}} \sigma_p}{\Gamma \rhd^{\inf} \mathsf{rec}\, x.\langle \varepsilon, \overline{a}_p\rangle.f' : \rho \dashv^{\mathsf{d}} \Sigma_{i\in I}\, a_i.\sigma_i}\ (p \in I)$$

$$(\text{Cpl}\oplus\text{-R}):\ \frac{\Gamma, x{:}\,\oplus_{i\in I}\, \overline{a}_i.\rho_i \dashv^{\mathsf{d}} \oplus_{j\in J}\, \overline{b}_j.\sigma_j \rhd^{\inf} f_j : \oplus_{i\in I}\, \overline{a}_i.\rho_i \dashv^{\mathsf{d}} \sigma_j \quad (\forall j \in J)}{\Gamma \rhd^{\inf} \mathsf{rec}\, x.\bigvee_{j\in J}\langle \varepsilon, b_j\rangle.f_j : \oplus_{i\in I}\, \overline{a}_i.\rho_i \dashv^{\mathsf{d}} \oplus_{j\in J}\, \overline{b}_j.\sigma_i}$$

$$(\text{Cpl}\oplus\text{-L}):\ \frac{\Gamma, x{:}\,\oplus_{i\in I}\, \overline{a}_i.\rho_i \dashv^{\mathsf{d}} \oplus_{j\in J}\, \overline{b}_j.\sigma_i \rhd^{\inf} f_i : \rho_i \dashv^{\mathsf{d}} \oplus_{j\in J}\, \overline{b}_j.\sigma_i \quad (\forall i \in I)}{\Gamma \rhd^{\inf} \mathsf{rec}\, x.\bigvee_{i\in I}\langle a_i, \varepsilon\rangle.f_i : \oplus_{i\in I}\, \overline{a}_i.\rho_i \dashv^{\mathsf{d}} \oplus_{j\in J}\, \overline{b}_j.\sigma_i}$$

$$(\text{Cpl}\oplus\text{-}\Sigma):\ \frac{\Gamma' \rhd^{\inf} f_i : \rho_i \dashv^{\mathsf{d}} \Sigma_{j\in J}\, a_j.\sigma_j\ (\forall i \in H) \quad \Gamma' \rhd^{\inf} f_i : \rho_i \dashv^{\mathsf{d}} \sigma_i\ (\forall i \in K)}{\Gamma \rhd^{\inf} f : \oplus_{i\in I}\, \overline{a}_i.\rho_i \dashv^{\mathsf{d}} \Sigma_{j\in J}\, a_j.\sigma_j}\ \left(\begin{smallmatrix} I = H \cup K, \\ K \subseteq J \end{smallmatrix}\right)$$

where $\Gamma' = \Gamma,\ x{:}\,\oplus_{i\in I}\, \overline{a}_i.\rho_i \dashv^{\mathsf{d}} \Sigma_{j\in J}\, a_j.\sigma_j$.
and $f = \mathsf{rec}\, x.(\bigvee_{h\in H}\langle a_h, \varepsilon\rangle.f_h) \vee (\bigvee_{k\in K}\langle a_k, \overline{a}_k\rangle.f_k)$

$$(\text{Cpl}\Sigma\text{-}\oplus):\ \frac{\Gamma' \rhd^{\inf} f_j : \Sigma_{i\in I}\, a_i.\rho_i \dashv^{\mathsf{d}} \sigma_j\ (\forall j \in H) \quad \Gamma' \rhd^{\inf} f_j : \rho_j \dashv^{\mathsf{d}} \sigma_j\ (\forall j \in K)}{\Gamma \rhd^{\inf} f : \Sigma_{i\in I}\, a_i.\rho_i \dashv^{\mathsf{d}} \oplus_{j\in J}\, \overline{a}_j.\sigma_j}\ \left(\begin{smallmatrix} J = H \cup K, \\ K \subseteq I \end{smallmatrix}\right)$$

where $\Gamma' = \Gamma, x{:}\,\Sigma_{i\in I}\, \overline{a}_i.\rho_i \dashv^{\mathsf{d}} \oplus_{j\in J}\, a_j.\sigma_j$
and $f = \mathsf{rec}\, x.(\bigvee_{h\in H}\langle \varepsilon, a_h\rangle.f_h) \vee (\bigvee_{k\in K}\langle \overline{a}_k, a_k\rangle.f_k)$

Fig. C.8. The inference system $\rhd^{\inf}$.

(CplΣ-L). In case it is possible to show that f' is an orchestrator for $f_p \dashv\!\vdash^{\mathsf{d}} \sigma$, orchestrated compliance can be obtained for $\Sigma_{i\in I}\, a_i.\rho_i \dashv\!\vdash^{\mathsf{d}} \sigma$ by means of $\langle \overline{a}_p, \varepsilon\rangle.f'$, since the $\langle \overline{a}_p, \varepsilon\rangle$ action satisfies one of the *input requests* a_is. In case $x \notin fn(f')$, we get that $\mathsf{rec}\, x.\langle \overline{a}_p, \varepsilon\rangle.f' = \langle \overline{a}_p, \varepsilon\rangle.f'$. This means that axiom (Ax) has been used in the derivation of f' and the interaction between $\Sigma_{i\in I}\, a_i.\rho_i$ and σ finitely succeeds if the actions described in the branch from (CplΣ-L) to (Ax) are performed. In case $x \in fn(f')$, rule (Hyp) has been used for f', and a successful infinite interaction is possible between $\Sigma_{i\in I}\, a_i.\rho_i$ and σ when the orchestrator repeatedly performs the actions in the branch from (CplΣ-L) to (Hyp), as described by the recursive orchestrator $\mathsf{rec}\, x.\langle \overline{a}_p, \varepsilon\rangle.f'$.

Proposition C.2 *($\rhd$-$\rhd^{\inf}$ equivalence). Let $f \in \mathsf{Orch}$ and $\rho, \sigma \in \mathsf{SC}$.*

$$\emptyset \rhd f : \rho \dashv^{\mathsf{d}} \sigma \ \text{ iff } \emptyset \rhd^{\inf} f : \rho \dashv^{\mathsf{d}} \sigma$$

Proof. The difference among the systems $\rhd$ and $\rhd^{\inf}$ primarily relies on the form of assumptions on the left hand side of the judgements, which are $f : \rho \dashv^{\mathsf{d}} \sigma$, with f a (closed) orchestrator in case of $\rhd$, and $x : \rho \dashv^{\mathsf{d}} \sigma$ in case of $\rhd^{\inf}$. Consequently the respective axioms (Hyp) are

$$\frac{}{\Gamma, f : \rho \dashv^{\mathsf{d}} \sigma \rhd f : \rho \dashv^{\mathsf{d}} \sigma} \qquad \frac{}{\Gamma, x : \rho \dashv^{\mathsf{d}} \sigma \rhd^{\inf} x : \rho \dashv^{\mathsf{d}} \sigma}$$

and whenever $\rhd$ has a coinductive rule of the shape

$$\frac{\Gamma, \odot_{i\in I}\, \mu_i.f_i : \rho \dashv^{\mathsf{d}} \sigma \rhd f_i : \rho_i \dashv^{\mathsf{d}} \sigma_i \quad (\forall i \in I)}{\Gamma \rhd \odot_{i\in I}\, \mu_i.f_i : \rho \dashv^{\mathsf{d}} \sigma}$$

for some operator $\odot$, in $\rhd^{\inf}$ there is a corresponding rule

$$\frac{\Gamma, x : \rho \dashv^{\mathsf{d}} \sigma \rhd^{\inf} f_i' : \rho_i \dashv^{\mathsf{d}} \sigma_i \quad (\forall i \in I)}{\Gamma \rhd^{\inf} \mathsf{rec}\, x.\odot_{i\in I}\, \mu_i.f_i' : \rho \dashv^{\mathsf{d}} \sigma}$$

where $f_i = f_i'\{\mathsf{rec}\, x.\odot_{i\in I}\, \mu_i.f_i'/x\}$.

Therefore to translate a derivation of system $\rhd^{\inf}$ into a derivation of system $\rhd$ it suffices to visit the derivation tree from the root, namely the conclusion, up to the leafs, by replacing each orchestrator $\mathsf{rec}\, x.f$ found on the right hand side of $\rhd^{\inf}$ in the conclusion of a rule as well as the assumption x in the premises of the same rule by $f\{\mathsf{rec}\, x.f/x\}$.

Vice versa to translate a derivation of $\rhd$ into one of $\rhd^{\inf}$ we can proceed in two passes. In the first pass we eliminate from the derivation all assumptions $f : \rho \dashv^{\mathrm{d}} \sigma$ which are not the right-hand side of a conclusion of (Hyp).

In the second pass if $\Gamma_k, g_k : \rho_k \dashv^{\mathrm{d}} \sigma_k \rhd g_k : \rho_k \dashv^{\mathrm{d}} \sigma_k$ for $k = 1, \ldots, n$ are all the conclusions of axiom (Hyp) in the given derivation, we replace each g_k by some new variable x_k. Then we observe that, but in the case of (Ax) which is the same in both systems, all rules in system $\rhd$ introduce an orchestrator $\odot_{i\in I}\, \mu_i . f_i$ with the f_i occurring on the right hand side of the premises. Then going from the leafs to the root of the derivation tree, we propagate the replacements of the g_k's by x_k's until $g_k : \rho_k \dashv^{\mathrm{d}} \sigma_k$ is the discharged assumption of a coinductive rule with the orchestrator $g_k = \odot_{i\in I}\, \mu_i . f_i$ in the conclusion: in that rule we replace g_k by x_k in the premises, and $\mathsf{rec}\, x . g'_k$ in the conclusion, where $g_k = g'_k\{\mathsf{rec}\, x_k . g'_k / x\}$.

A routine induction proves that these translations are correct, sending derivations of one system into those of the other one, where the orchestrators in the conclusion are equivalent up to unfolding. □

Appendix D. Proof of Proposition 5.4

We shall first prove Proposition 5.4(2), whereas the proof of 5.4(3) will follow as a corollary. The proof of

$$\rho \dashv\!\vdash \sigma \text{ and } \overline{\sigma} \dashv\!\vdash \sigma' \implies \rho \dashv\!\vdash \sigma'$$

will proceed as follows. We first show that, given f and g such that $f : \rho \dashv\!\vdash^{\mathrm{d}} \sigma$ and $g : \overline{\sigma} \dashv\!\vdash^{\mathrm{d}} \sigma'$, it is possible to build an orchestrator h such that $h : \rho \dashv\!\vdash^{\mathrm{d}} \sigma'$ (the building procedure for h resembles a similar procedure of [11]). The orchestrator h coordinates the orchestration actions of f and g, fusing together the buffers of f and g. For instance it transforms an action $\langle a, \overline{a}\rangle$ of f and an action $\langle a, \varepsilon\rangle$ of g in an action $\langle a, \varepsilon\rangle$. Two actions $\langle \varepsilon, \overline{a}\rangle$ and $\langle a, \varepsilon\rangle$, respectively of f and g, when coordinated together, do annihilate each other, producing no action at all (like taking an element from a buffer and immediately putting it back).

We shall build h out of the two derivations of the judgements $\rhd^{\inf} f : \rho \dashv^{\mathrm{d}} \sigma$ and $\rhd^{\inf} g : \overline{\sigma} \dashv^{\mathrm{d}} \sigma'$ which do exist because of the soundness and completeness of system $\rhd^{\inf}$ with respect to the relation $\dashv\!\vdash^{\mathrm{d}}$. What we actually do is get h out of a proof reconstruction procedure for $\rhd^{\inf} h : \rho \dashv^{\mathrm{d}} \sigma'$ that shall be proved to be terminating. The orchestrator h can also be looked at as the result of a sort of orchestrator-composition operator, i,e, $h = f \bullet g$. Of course in order to prove Proposition 5.4(2) out of the result

$$f : \rho \dashv\!\vdash^{\mathrm{d}} \sigma \;\&\; g : \overline{\sigma} \dashv\!\vdash^{\mathrm{d}} \sigma' \implies (f \bullet g) : \rho \dashv\!\vdash^{\mathrm{d}} \sigma'$$

we have to show that the operation $\cdot\bullet\cdot$ does preserve the respectfulness of orchestrators, since the relation $\dashv\!\vdash$ is defined in terms of the existence of *respectful orchestrators* for $\dashv\!\vdash^{\mathrm{d}}$.

We start by defining a recursive procedure **P** with three arguments: two derivations in the formal system $\rhd^{\inf}$ and one environment.

Definition D.1 *(The algorithm* **P***).* The algorithm $\mathbf{P}\,(\mathcal{D}_1, \mathcal{D}_2, \Gamma)$, where $\mathcal{D}_1$ and $\mathcal{D}_2$ are derivations in $\rhd^{\inf}$ and Γ is an environment, is defined by providing the defining clauses according to the last rules in the derivations $\mathcal{D}_1$ and $\mathcal{D}_2$. We name the clauses according to the name of the last rules of the two derivations. In names like R-$*$, the symbol $*$ stands for any rule that can be paired with R.

We assume that the application priority of the rules respect the order in which they are listed below.

Clause Init : The computation of

$$\mathbf{P}\,\big(\mathcal{D}_1 :: \Gamma_1 \rhd^{\inf} f : \rho \dashv^{\mathrm{d}} \sigma, \mathcal{D}_2 :: \Gamma_2 \rhd^{\inf} g : \overline{\sigma} \dashv^{\mathrm{d}} \sigma', \Gamma_3, x : \rho \dashv^{\mathrm{d}} \sigma'\big)$$

consists in simply returning

$$(\textsc{Hyp}) :: \Gamma_3, x : \rho \dashv^{\mathrm{d}} \sigma' \rhd^{\inf} x : \rho \dashv^{\mathrm{d}} \sigma'.$$

Clause (**Ax**)$-*$: $\mathbf{P}\,\big((\textsc{Ax}) :: \Gamma_1 \rhd^{\inf} 1 : \mathbf{1} \dashv^{\mathrm{d}} \sigma,\ \mathcal{D}_2 :: \Gamma_2 \rhd^{\inf} g : \overline{\sigma} \dashv^{\mathrm{d}} \sigma',\ \Gamma_3\big)$.

We return $(\textsc{Ax}) :: \Gamma_3 \rhd^{\inf} 1 : \mathbf{1} \dashv^{\mathrm{d}} \sigma'$.

Notice that it is not necessary to have a **Clause** $*$-(**Ax**), since in that case $\sigma = \mathbf{1}$ and hence also $\rho = \mathbf{1}$. This means that **Clause** (**Ax**)-$*$ applies.

Clause (**Hyp**)-(**Hyp**) :

$$\mathbf{P}\,\big((\textsc{Hyp}) :: \Gamma_1, x : \rho \dashv^{\mathrm{d}} \sigma \rhd^{\inf} x : \rho \dashv^{\mathrm{d}} \sigma,\ (\textsc{Hyp}) :: \Gamma_2, x : \overline{\sigma} \dashv^{\mathrm{d}} \sigma' \rhd^{\inf} x : \overline{\sigma} \dashv^{\mathrm{d}} \sigma',\ \Gamma_3\big).$$

Let now $\mathcal{D}_1 :: \Gamma_1 \rhd^{\inf} \mathsf{rec}\, x . f' : \rho \dashv^{\mathrm{d}} \sigma$ be the subderivation (of the initial derivation) where the variable x is discharged, and similarly for $\mathcal{D}_2 :: \Gamma_2 \rhd^{\inf} \mathsf{rec}\, x . g' : \overline{\sigma} \dashv^{\mathrm{d}} \sigma'$. Then we return

$$\mathbf{P}\,\big(\mathcal{D}_1 :: \Gamma_1 \rhd^{\inf} \mathsf{rec}\, x . f' : \rho \dashv^{\mathrm{d}} \sigma,\ \mathcal{D}_2 :: \Gamma_2 \rhd^{\inf} \mathsf{rec}\, x . g' : \overline{\sigma} \dashv^{\mathrm{d}} \sigma',\ \Gamma_3\big)$$

Clause (**Hyp**)-$*$: $\mathbf{P}\left((\text{Hyp}) :: \Gamma_1, x:\rho \dashv^{\mathsf{d}} \sigma \rhd^{\inf} x:\rho \dashv^{\mathsf{d}} \sigma,\ \mathcal{D}_2 :: \Gamma_2 \rhd^{\inf} g:\overline{\sigma} \dashv^{\mathsf{d}} \sigma',\ \Gamma_3\right)$.

Let now $\mathcal{D}_1 :: \Gamma_1 \rhd^{\inf} \mathsf{rec}x.f' : \rho \dashv^{\mathsf{d}} \sigma$ be the subderivation (of the initial derivation) where the variable x is discharged. Then we return

$$\mathbf{P}\left(\mathcal{D}_1 :: \Gamma_1 \rhd^{\inf} \mathsf{rec}x.f' : \rho \dashv^{\mathsf{d}} \sigma,\ \mathcal{D}_2 :: \Gamma_2 \rhd^{\inf} g:\overline{\sigma} \dashv^{\mathsf{d}} \sigma',\ \Gamma_3\right)$$

Clause $*$-(**Hyp**) : $\mathbf{P}\left(\mathcal{D}_1 :: \Gamma_1 \rhd^{\inf} f:\rho \dashv^{\mathsf{d}} \sigma,\ (\text{Hyp}) :: \Gamma_2, x:\overline{\sigma} \dashv^{\mathsf{d}} \sigma' \rhd^{\inf} x:\overline{\sigma} \dashv^{\mathsf{d}} \sigma',\ \Gamma_3\right)$.

Let now $\mathcal{D}_2 :: \Gamma_2 \rhd^{\inf} \mathsf{rec}x.g' : \overline{\sigma} \dashv^{\mathsf{d}} \sigma'$ be the subderivation (of the initial derivation) where the variable x is discharged. Then we return

$$\mathbf{P}\left(\mathcal{D}_1 :: \Gamma_1 \rhd^{\inf} f:\rho \dashv^{\mathsf{d}} \sigma,\ \mathcal{D}_2 :: \Gamma_2 \rhd^{\inf} \mathsf{rec}x.g' : \overline{\sigma} \dashv^{\mathsf{d}} \sigma',\ \Gamma_3\right)$$

Notice that no rule among the last three above can be applied immediately after the application of one of them.

Clause (**CPLΣ-L**)-$*$: $\mathbf{P}\left(\mathcal{D}'_1, \mathcal{D}_2 :: \Gamma_2 \rhd^{\inf} g:\overline{\sigma} \dashv^{\mathsf{d}} \sigma', \Gamma_3\right)$ where

$$\mathcal{D}'_1 = \dfrac{\boxed{\mathcal{D}_1}}{\dfrac{\Gamma'_1 \rhd^{\inf} f' : \rho_p \dashv^{\mathsf{d}} \sigma}{\Gamma_1 \rhd^{\inf} \mathsf{rec}x.\langle \overline{a}_p, \varepsilon\rangle.f' : \Sigma_{i\in I} a_i.\rho_i \dashv^{\mathsf{d}} \sigma}\ (p \in I)}$$

and $\Gamma'_1 = \Gamma_1, x:\Sigma_{i\in I} a_i.\rho_i \dashv^{\mathsf{d}} \sigma$.

Let $\mathbf{P}\left(\mathcal{D}_1 :: \Gamma'_1 \rhd^{\inf} f':\rho_p \dashv^{\mathsf{d}} \sigma, \mathcal{D}_2 :: \Gamma_2 \rhd^{\inf} g:\overline{\sigma} \dashv^{\mathsf{d}} \sigma', \Gamma_3, x:\rho \dashv^{\mathsf{d}} \sigma\right) = \mathcal{D} :: \Gamma_3, x:\rho \dashv^{\mathsf{d}} \sigma \rhd h:\rho_p \dashv^{\mathsf{d}} \sigma'$. Then we return

$$\dfrac{\boxed{\mathcal{D}}}{\dfrac{\Gamma_3, x:\rho \dashv^{\mathsf{d}} \sigma \rhd^{\inf} h:\rho_p \dashv^{\mathsf{d}} \sigma'}{\Gamma_3 \rhd^{\inf} \mathsf{rec}x.\langle \overline{a}_p, \varepsilon\rangle.h : \rho \dashv^{\mathsf{d}} \sigma'}\ (\text{CPL}\Sigma\text{-L})}$$

Clause (**CPL$\oplus$-L**)-$*$: $\mathbf{P}\left(\mathcal{D}'_1, \mathcal{D}_2 :: \Gamma_2 \rhd g:\overline{\sigma} \dashv^{\mathsf{d}} \sigma', \Gamma_3\right)$ where

$$\mathcal{D}'_1 = \dfrac{\boxed{\mathcal{D}_i}}{\dfrac{\Gamma'_1 \rhd f_i : \rho_i \dashv^{\mathsf{d}} \sigma \quad (\forall i \in I)}{\Gamma_1 \rhd \bigvee_{i\in I}\langle a_i, \varepsilon\rangle.f_i : \oplus_{i\in I} \overline{a}_i.\rho_i \dashv^{\mathsf{d}} \sigma}\ (\text{CPL}\oplus\text{-L})}$$

and $\Gamma'_1 = \Gamma, \bigvee_{i\in I}\langle a_i, \varepsilon\rangle.f_i : \oplus_{i\in I} \overline{a}_i.\rho_i \dashv^{\mathsf{d}} \sigma$.

Let $\mathbf{P}\left(\mathcal{D}_i :: \Gamma'_1 \rhd^{\inf} f_i : \rho_i \dashv^{\mathsf{d}} \sigma, \mathcal{D}_2 :: \Gamma' \rhd g:\overline{\sigma} \dashv^{\mathsf{d}} \sigma', \Gamma_3, x:\rho \dashv^{\mathsf{d}} \sigma'\right) = \mathcal{D}'_i :: \Gamma_3, x:\rho \dashv^{\mathsf{d}} \sigma' \rhd h_i : \rho_i \dashv^{\mathsf{d}} \sigma'$. We then return

$$\dfrac{\boxed{\mathcal{D}'_i}}{\dfrac{\Gamma_3, x:\rho \dashv^{\mathsf{d}} \sigma' \rhd h_i : \rho_i \dashv^{\mathsf{d}} \sigma' \quad (\forall i \in I)}{\Gamma_3 \rhd \bigvee_{i\in I}\langle a_i, \varepsilon\rangle.h_i : \oplus_{i\in I} \overline{a}_i.\rho_i \dashv^{\mathsf{d}} \sigma'}\ (\text{CPL}\oplus\text{-L})}$$

Clause $*$-(**CPLΣ-R**) : $\mathbf{P}\left(\mathcal{D}_1 :: \Gamma'_1 \rhd^{\inf} f' : \rho \dashv^{\mathsf{d}} \sigma,\ \mathcal{D}'_2,\ \Gamma_3\right)$ where

$$\mathcal{D}'_2 = \dfrac{\boxed{\mathcal{D}_2}}{\dfrac{\Gamma'_2 \rhd^{\inf} g' : \overline{\sigma} \dashv^{\mathsf{d}} \sigma'_p}{\Gamma_1 \rhd^{\inf} \mathsf{rec}x.\langle \varepsilon, \overline{a}_p\rangle.g' : \overline{\sigma} \dashv^{\mathsf{d}} \Sigma_{i\in I} a_i.\sigma'_i}\ (p \in I)}$$

and $\Gamma'_2 = \Gamma_1, x:\overline{\sigma} \dashv^{\mathsf{d}} \Sigma_{i\in I} a_i.\sigma'_i$.

Let $\mathbf{P}\left(\mathcal{D}_1 :: \Gamma'_1 \rhd^{\inf} f' : \rho \dashv^{\mathsf{d}} \sigma, \mathcal{D}_2 :: \Gamma' \rhd^{\inf} g' : \overline{\sigma} \dashv^{\mathsf{d}} \sigma'_p, \Gamma_3, x:\rho \dashv^{\mathsf{d}} \sigma'\right) = \mathcal{D}' :: \Gamma_3, x:\rho \dashv^{\mathsf{d}} \sigma' \rhd h:\rho \dashv^{\mathsf{d}} \sigma'_p$. We then return

$$\dfrac{\boxed{\mathcal{D}'}}{\dfrac{\Gamma_3, x:\rho \dashv^{\mathsf{d}} \sigma' \rhd^{\inf} h:\overline{\sigma} \dashv^{\mathsf{d}} \sigma'_p \quad (p \in I)}{\Gamma_3 \rhd^{\inf} \mathsf{rec}x.\langle \varepsilon, \overline{a}_p\rangle.h : \overline{\sigma} \dashv^{\mathsf{d}} \Sigma_{i\in I} a_i.\sigma'_i}\ (\text{CPL}\Sigma\text{-R})}$$

Clause (CPLΣ-**R**)-(CPL$\oplus$-**L**) : $\mathbf{P}\left(\mathcal{D}'_1,\ \mathcal{D}'_2,\ \Gamma_3\right)$ where

$$\mathcal{D}'_1 = \dfrac{\boxed{\mathcal{D}_1}}{\dfrac{\Gamma'_1 \rhd^{\inf} f' : \rho \dashv^{\mathsf{d}} \sigma_p \quad (p \in I)}{\Gamma_1 \rhd^{\inf} \mathsf{rec}\, x.\langle \varepsilon, \overline{a}_p\rangle.f' : \rho \dashv^{\mathsf{d}} \Sigma_{i\in I}\, a_i.\sigma_i}\ (\text{CPL}\Sigma\text{-R})}$$

$$\mathcal{D}'_2 = \dfrac{\boxed{\mathcal{D}^i_2}}{\dfrac{\Gamma'_2 \rhd g_i : \overline{\sigma}_i \dashv^{\mathsf{d}} \sigma' \quad (\forall i \in I)}{\Gamma_2 \rhd \bigvee_{i\in I} \langle a_i, \varepsilon\rangle.g_i : \oplus_{i\in I}\, \overline{a}_i.\overline{\sigma}_i \dashv^{\mathsf{d}} \sigma'}\ (\text{CPL}\oplus\text{-L})}$$

$\Gamma'_1 = \Gamma_1, x : \Sigma_{i\in I}\, \overline{a}_i.\overline{\sigma}_i \dashv^{\mathsf{d}} \sigma'$, and $\Gamma'_2 = \Gamma_2, x : \oplus_{i\in I}\, \overline{a}_i.\overline{\sigma}_i \dashv^{\mathsf{d}} \sigma'$. Let

$$\mathbf{P}\left(\mathcal{D}_1 :: \Gamma'_1 \rhd^{\inf} f' : \rho \dashv^{\mathsf{d}} \sigma_p, \mathcal{D}^p_2 :: \Gamma' \rhd g_p : \overline{\sigma}_p \dashv^{\mathsf{d}} \sigma', \Gamma_3\right) = \mathcal{D} :: \Gamma_3 \rhd h : \rho \dashv^{\mathsf{d}} \sigma'$$

We then return $\mathcal{D}$.

(Notice that here, in the recursive calls, ρ and σ' remain unchanged. That is why we do not add $x : \rho \dashv^{\mathsf{d}} \sigma'$ to Γ_3. However, as we shall show in the proof of termination of the algorithm, there cannot be an infinite sequence of consecutive applications of clauses like the present one.)

Clause (CPLΣ-**R**)-(CPL$\oplus$-Σ) : $\mathbf{P}\left(\mathcal{D}'_1, \mathcal{D}'_2, \Gamma_3\right)$ where

$$\mathcal{D}'_1 = \dfrac{\boxed{\mathcal{D}_1}}{\dfrac{\Gamma'_1 \rhd^{\inf} f' : \rho \dashv^{\mathsf{d}} \sigma_p \quad (p \in I)}{\Gamma_1 \rhd^{\inf} \mathsf{rec}\, x.\langle \varepsilon, \overline{a}_p\rangle.f' : \rho \dashv^{\mathsf{d}} \Sigma_{i\in I}\, a_i.\sigma_i}\ (\text{CPL}\Sigma\text{-R})}$$

$$\mathcal{D}'_2 = \dfrac{\boxed{\mathcal{D}_i} \quad \boxed{\mathcal{D}_i}}{\dfrac{\Gamma'_2 \rhd g_i : \overline{\sigma}_i \dashv^{\mathsf{d}} \Sigma_{j\in J}\, a_j.\sigma'_j \quad (\forall i \in H) \qquad \Gamma'_2 \rhd g_i : \overline{\sigma}_i \dashv^{\mathsf{d}} \sigma'_i \quad (\forall i \in K)}{\Gamma_2 \rhd (\bigvee_{h\in H} \langle a_h, \varepsilon\rangle.g_h) \vee (\bigvee_{k\in K} \langle a_k, \overline{a}_k\rangle.g_k) : \oplus_{i\in H\cup K}\, \overline{a}_i.\overline{\sigma}_i \dashv^{\mathsf{d}} \Sigma_{j\in J}\, a_j.\sigma'_j}\ (\text{CPL}\oplus\text{-}\Sigma)}$$

in $\mathcal{D}'_1$, $\Gamma'_1 = \Gamma_1, x : \rho \dashv^{\mathsf{d}} \Sigma_{i\in I}\, a_i.\sigma_i$ and in $\mathcal{D}'_2$, $K \subseteq J$, $H \cup K = I$ and $\Gamma'_2 = \Gamma_2, x : \oplus_{i\in(H\cup K)}\, \overline{a}_i.\overline{\sigma}_i \dashv^{\mathsf{d}} \Sigma_{j\in J}\, a_j.\sigma'_j$.

We distinguish two cases:

$p \in H$: Let $\mathbf{P}\left(\mathcal{D}_1 :: \Gamma'_1 \rhd^{\inf} f' : \rho \dashv^{\mathsf{d}} \sigma_p,\ \mathcal{D}_p :: \Gamma'_2 \rhd g_p : \overline{\sigma}_p \dashv^{\mathsf{d}} \Sigma_{j\in J}\, a_j.\sigma'_j, \Gamma_3\right) = \mathcal{D} :: \Gamma_3 \rhd h : \rho \dashv^{\mathsf{d}} \sigma'$.

We then return

$$\mathcal{D} :: \Gamma_3 \rhd^{\inf} h : \rho \dashv^{\mathsf{d}} \sigma'$$

(The same observation at the end of the previous clause applies here.)

$p \in K$: Let $\mathbf{P}\left(\mathcal{D}_1 :: \Gamma'_1 \rhd^{\inf} f' : \rho \dashv^{\mathsf{d}} \sigma_p,\ \mathcal{D}_p :: \Gamma'_2 \rhd g_p : \overline{\sigma}_p \dashv^{\mathsf{d}} \sigma'_p\ ,\ \Gamma_3, x : \rho \dashv^{\mathsf{d}} \sigma'\right) = \mathcal{D} :: \Gamma_3, x : \rho \dashv^{\mathsf{d}} \sigma' \rhd h : \rho \dashv^{\mathsf{d}} \sigma_p$. We then return

$$\dfrac{\boxed{\mathcal{D}}}{\dfrac{\Gamma_3, x : \rho \dashv^{\mathsf{d}} \Sigma_{j\in J}\, a_j.\sigma'_j \rhd^{\inf} h : \rho \dashv^{\mathsf{d}} \sigma'_p}{\Gamma_3 \rhd^{\inf} \mathsf{rec}\, x.\langle \varepsilon, \overline{a}_p\rangle.h : \rho \dashv^{\mathsf{d}} \Sigma_{j\in J}\, a_j.\sigma'_j}\ (\text{CPL}\Sigma\text{-R})}$$

Clause (CPL$\oplus$-**R**)-(CPLΣ-**L**) : $\mathbf{P}\left(\mathcal{D}_1, \mathcal{D}_2, \Gamma_3\right)$ where

$$\mathcal{D}_1 = \dfrac{\boxed{\mathcal{D}_i}}{\dfrac{\Gamma'_1 \rhd f_i : \rho \dashv^{\mathsf{d}} \sigma_i \quad (\forall i \in I)}{\Gamma_1 \rhd \bigvee_{i\in I} \langle \varepsilon, a_i\rangle.f_i : \rho \dashv^{\mathsf{d}} \oplus_{i\in I}\, \overline{a}_i.\sigma_i}\ (\text{CPL}\oplus\text{-R})}$$

$$\mathcal{D}_2 = \dfrac{\boxed{\mathcal{D}'_p}}{\dfrac{\Gamma'_2 \rhd g' : \overline{\sigma}_p \dashv^{\mathsf{d}} \sigma' \quad (p \in I)}{\Gamma_2 \rhd \langle \overline{a}_p, \varepsilon\rangle.g' : \Sigma_{i\in I}\, a_i.\overline{\sigma}_i \dashv^{\mathsf{d}} \sigma'}\ (\text{CPL}\Sigma\text{-L})}$$

$\Gamma'_1 = \Gamma_1, x : \rho \dashv^{\mathsf{d}} \oplus_{i\in I}\, \overline{a}_i.\sigma_i$, $\Gamma'_2 = \Gamma_2, x : \Sigma_{i\in I}\, a_i.\overline{\sigma}_i \dashv^{\mathsf{d}} \sigma'$.

Let $\mathbf{P}\left(\mathcal{D}_p :: \Gamma'_1 \rhd^{\mathsf{inf}} f_p : \rho \dashv^{\mathsf{d}} \sigma_p,\ \mathcal{D}'_p :: \Gamma'_2 \rhd g' : \overline{\sigma}_p \dashv^{\mathsf{d}} \sigma'\ ,\ \Gamma_3\right) = \mathcal{D} :: \Gamma_3 \rhd h : \rho \dashv^{\mathsf{d}} \sigma'$. We then return $\mathcal{D}$.

Clause (CPL⊕-**R**)-(CPLΣ-⊕) : $\mathbf{P}\left(\mathcal{D}_1, \mathcal{D}_2, \Gamma_3\right)$ where

$$\mathcal{D}_1 = \dfrac{\boxed{\mathcal{D}_i}\quad \Gamma'_1 \rhd f_i : \rho \dashv^{\mathsf{d}} \sigma_i \quad (\forall i \in I)}{\Gamma_1 \rhd \bigvee_{i\in I}\langle \varepsilon, a_i\rangle.f_i : \rho \dashv^{\mathsf{d}} \oplus_{i\in I} \overline{a}_i.\sigma_i}\ (\text{CPL}\oplus\text{-R})$$

$$\mathcal{D}_2 = \dfrac{\boxed{\mathcal{D}'_j}\quad \Gamma'_2 \rhd g_j : \Sigma_{i\in I}\, a_i.\overline{\sigma}_i \dashv^{\mathsf{d}} \sigma'_j\ (\forall j \in H) \qquad \boxed{\mathcal{D}'_j}\quad \Gamma'_2 \rhd g_j : \overline{\sigma}_j \dashv^{\mathsf{d}} \sigma'_j\ (\forall j \in K) \quad (K \subseteq I)}{\Gamma_2 \rhd \bigvee_{i\in H}\langle \varepsilon, a_i\rangle.g_i) \vee (\bigvee_{i\in K}\langle \overline{a}_i, a_i\rangle.g_i : \Sigma_{i\in I}\, a_i.\overline{\sigma}_i \dashv^{\mathsf{d}} \oplus_{j\in H\cup K} \overline{a}_j.\sigma'_j}\ (\text{CPL}\Sigma\text{-}\oplus)$$

$\Gamma'_1 = \Gamma_1, x : \rho \dashv^{\mathsf{d}} \oplus_{i\in I} \overline{a}_i.\sigma_i$, and $\Gamma'_2 = \Gamma_2, x : \Sigma_{i\in I}\, a_i.\overline{\sigma}_i \dashv^{\mathsf{d}} \oplus_{j\in H\cup K} \overline{a}_j.\sigma'_j$.

Let, for all $i \in K$, $\mathbf{P}\big(\mathcal{D}_i :: \Gamma'_1 \rhd f_i : \rho \dashv^{\mathsf{d}} \sigma_i,\ \mathcal{D}'_i :: \Gamma'_2 \rhd g_i : \overline{\sigma}_i \dashv^{\mathsf{d}} \sigma'_i, \Gamma_3, x : \rho \dashv^{\mathsf{d}} \sigma'\big) = \mathcal{D}''_i :: \Gamma_3, x : \rho \dashv^{\mathsf{d}} \sigma' \rhd h_i : \rho \dashv^{\mathsf{d}} \sigma_i$, and, for all $j \in H$ $\mathbf{P}\left(\mathcal{D}', \mathcal{D}'_j :: \Gamma'_2 \rhd g_j : \Sigma_{i\in I}\, a_i.\overline{\sigma}_i \dashv^{\mathsf{d}} \sigma'_j\ , \Gamma_3, x : \rho \dashv^{\mathsf{d}} \sigma'\right) = \mathcal{D}''_j :: \Gamma_3, x : \rho \dashv^{\mathsf{d}} \sigma' \rhd h_j : \rho \dashv^{\mathsf{d}} \sigma_j$, where

$$\mathcal{D}' = \dfrac{\boxed{\mathcal{D}_i}\quad \Gamma'_1 \rhd f_i : \rho \dashv^{\mathsf{d}} \sigma_i \quad (\forall i \in I)}{\Gamma_1 \rhd \bigvee_{i\in I}\langle \varepsilon, a_i\rangle.f_i : \rho \dashv^{\mathsf{d}} \oplus_{i\in I} \overline{a}_i.\sigma_i}$$

We then return

$$\dfrac{\boxed{\mathcal{D}''_i}\quad \Gamma_3, x : \rho \dashv^{\mathsf{d}} \sigma' \rhd h_i : \rho \dashv^{\mathsf{d}} \sigma'_i \quad (\forall i \in H \cup K)}{\Gamma_3 \rhd \bigvee_{i\in H\cup K}\langle \varepsilon, a_i\rangle.h_i : \rho \dashv^{\mathsf{d}} \oplus_{i\in H\cup K} \overline{a}_i.\sigma'_i}\ (\text{CPL}\oplus\text{-R})$$

Clause (CPL⊕-Σ)-(CPL⊕-**L**) : $\mathbf{P}\left(\mathcal{D}_1, \mathcal{D}_2, \Gamma_3\right)$, where

$$\mathcal{D}_1 = \dfrac{\boxed{\mathcal{D}_i}\quad \Gamma'_1 \rhd f_i : \rho_i \dashv^{\mathsf{d}} \Sigma_{j\in J}\, a_j.\sigma_j\ (\forall i \in H) \quad \boxed{\mathcal{D}_i}\quad \Gamma'_1 \rhd f_i : \rho_i \dashv^{\mathsf{d}} \sigma_i\ (\forall i \in K) \quad (K \subseteq J)}{\Gamma_1 \rhd \bigvee_{h\in H}\langle a_h, \varepsilon\rangle.f_h) \vee (\bigvee_{k\in K}\langle a_k, \overline{a}_k\rangle.f_k : \oplus_{i\in H\cup K} \overline{a}_i.\rho_i \dashv^{\mathsf{d}} \Sigma_{j\in J}\, a_j.\sigma_j}\ (\text{CPL}\oplus\text{-}\Sigma)$$

$$\mathcal{D}_2 = \dfrac{\boxed{\mathcal{D}_j}\quad \Gamma'_2 \rhd g_j : \overline{\sigma}_j \dashv^{\mathsf{d}} \sigma' \quad (\forall j \in J)}{\Gamma_2 \rhd \bigvee_{j\in J}\langle a_j, \varepsilon\rangle.g_j : \oplus_{i\in J} \overline{a}_j.\overline{\sigma}_j \dashv^{\mathsf{d}} \sigma'}\ (\text{CPL}\oplus\text{-L})$$

where $\Gamma'_1 = \Gamma_1, f : \oplus_{i\in H\cup K} \overline{a}_i.\rho_i \dashv^{\mathsf{d}} \Sigma_{j\in J}\, a_j.\sigma_j$, and $\Gamma'_2 = \Gamma_2, \bigvee_{i\in I}\langle a_i, \varepsilon\rangle.g_i : \oplus_{i\in J} \overline{a}_j.\overline{\sigma}_j \dashv^{\mathsf{d}} \sigma'$.

Let, for all $i \in H$, $\mathbf{P}\big(\mathcal{D}_i :: \Gamma'_1 \rhd f_i : \rho_i \dashv^{\mathsf{d}} \Sigma_{j\in J}\, a_j.\sigma_j, \mathcal{D}'_2, \Gamma_3, x : \oplus_{i\in H\cup K} \overline{a}_i.\rho_i \dashv^{\mathsf{d}} \sigma'\big) = \mathcal{D}'_i :: \Gamma_3, x : \rho \dashv^{\mathsf{d}} \sigma' \rhd^{\mathsf{inf}} h_i : \rho_i \dashv^{\mathsf{d}} \sigma'$, where

$$\mathcal{D}'_2 = \dfrac{\boxed{\mathcal{D}_j}\quad \Gamma'_2 \rhd g_j : \overline{\sigma}_j \dashv^{\mathsf{d}} \sigma' \quad (\forall j \in J)}{\Gamma_2 \rhd \bigvee_{j\in J}\langle a_j, \varepsilon\rangle.g_j : \oplus_{i\in J} \overline{a}_j.\overline{\sigma}_j \dashv^{\mathsf{d}} \sigma'}$$

and let for all $i \in K$,
$\mathbf{P}\big(\mathcal{D}_i :: \Gamma'_1 \rhd f_i : \rho_i \dashv^{\mathsf{d}} \sigma_i, \mathcal{D}_i :: \Gamma'_2 \rhd g_i : \overline{\sigma}_i \dashv^{\mathsf{d}} \sigma', \Gamma_3, x : \oplus_{i\in H\cup K} \overline{a}_i.\rho_i \dashv^{\mathsf{d}} \Sigma_{j\in J}\, a_j.\sigma_j\big) = \mathcal{D}'_i :: \Gamma_3, x : \rho \dashv^{\mathsf{d}} \sigma' \rhd^{\mathsf{inf}} h_i : \rho_i \dashv^{\mathsf{d}} \sigma'$.

We then return, using rule (CPL⊕-L),

$$\dfrac{\boxed{\mathcal{D}'_i}\quad \Gamma_3, x : \rho \dashv^{\mathsf{d}} \sigma' \rhd^{\mathsf{inf}} h_i : \rho_i \dashv^{\mathsf{d}} \sigma'\ (\forall i \in H) \quad \boxed{\mathcal{D}'_i}\quad \Gamma_3, x : \rho \dashv^{\mathsf{d}} \sigma' \rhd^{\mathsf{inf}} h_i : \rho_i \dashv^{\mathsf{d}} \sigma'\ (\forall i \in K)}{\Gamma_3 \rhd \bigvee_{i\in H\cup K}\langle a_i, \varepsilon\rangle.h_i : \oplus_{i\in H\cup K} \overline{a}_i.\rho_i \dashv^{\mathsf{d}} \sigma'}\ (\text{CPL}\oplus\text{-L})$$

Clause (CPL⊕-Σ)-(CPL⊕-Σ) : **P** $\left(\mathcal{D}_1, \mathcal{D}_2, \Gamma_3\right)$ where

$$\mathcal{D}_1 = \dfrac{\boxed{\mathcal{D}_i} \quad \boxed{\mathcal{D}_i}}{\Gamma_1 \rhd \bigvee_{h\in H}\langle a_h, \varepsilon\rangle.f_h) \vee (\bigvee_{k\in K}\langle a_k, \overline{a}_k\rangle.f_k : \oplus_{i\in H\cup K}\overline{a}_i.\rho_i \dashv^{\mathsf{d}} \Sigma_{j\in J}\, a_j.\sigma_j} \; (\text{CPL}\oplus\text{-}\Sigma)$$

$$\mathcal{D}_2 = \dfrac{\boxed{\mathcal{D}_i} \quad \boxed{\mathcal{D}_i}}{\Gamma_2 \rhd \bigvee_{h\in H'}\langle a_h, \varepsilon\rangle.g_h) \vee (\bigvee_{k\in K'}\langle a_k, \overline{a}_k\rangle.g_k : \oplus_{i\in (H'\cup K')=J}\overline{a}_i.\overline{\sigma}_i \dashv^{\mathsf{d}} \Sigma_{j\in J'}\, a_j.\sigma'_j} \; (\text{CPL}\oplus\text{-}\Sigma)$$

$H' \cup K' = J$, $\Gamma'_1 = \Gamma_1$, $f : \oplus_{i\in H\cup K}\overline{a}_i.\rho_i \dashv^{\mathsf{d}} \Sigma_{j\in J}\, a_j.\sigma_j$, and $\Gamma'_2 = \Gamma_2$, $g : \oplus_{i\in (H'\cup K')=J}\overline{a}_i.\overline{\sigma}_i \dashv^{\mathsf{d}} \Sigma_{j\in J'}\, a_j.\sigma'_j$.

Let for all $i \in H$,

$\mathbf{P}\big(\mathcal{D}_i :: \Gamma'_1 \rhd f_i : \rho_i \dashv^{\mathsf{d}} \Sigma_{j\in J}\, a_j.\sigma_j,\ \mathcal{D}_2, \Gamma_3, x : \oplus_{i\in H\cup K}\overline{a}_i.\rho_i \dashv^{\mathsf{d}} \Sigma_{j\in J'}\, a_j.\sigma'_j\big) = \mathcal{D}'_i :: \Gamma_3, x : \rho \dashv^{\mathsf{d}} \sigma' \rhd^{\mathsf{inf}} h_i : \rho_i \dashv^{\mathsf{d}} \sigma'$,
where

$$\mathcal{D}_2 = \dfrac{\boxed{\mathcal{D}_i} \quad \boxed{\mathcal{D}_i}}{\Gamma_2 \rhd g : \oplus_{i\in (H'\cup K')=J}\overline{a}_i.\overline{\sigma}_i \dashv^{\mathsf{d}} \Sigma_{j\in J'}\, a_j.\sigma'_j}$$

and let for all $i \in H' \cap K$: $\mathbf{P}\big(\mathcal{D}_i :: \Gamma'_1 \rhd f_i : \rho_i \dashv^{\mathsf{d}} \sigma_i,\ \mathcal{D}_i :: \Gamma'_2 \rhd g_i : \overline{\sigma}_i \dashv^{\mathsf{d}} \Sigma_{j\in J'}\, a_j.\sigma'_j, \Gamma_3, x : \oplus_{i\in H\cup K}\overline{a}_i.\rho_i \dashv^{\mathsf{d}} \Sigma_{j\in J'}\, a_j.\sigma'_j\big) = \mathcal{D}'_i :: \Gamma_3, x : \rho \dashv^{\mathsf{d}} \sigma' \rhd^{\mathsf{inf}} h_i : \rho_i \dashv^{\mathsf{d}} \sigma'$,
and let for all $i \in K' \cap K$:
$\mathbf{P}\big(\mathcal{D}_i :: \Gamma'_1 \rhd f_i : \rho_i \dashv^{\mathsf{d}} \sigma_i,\ \mathcal{D}_i :: \Gamma'_2 \rhd g_i : \overline{\sigma}_i \dashv^{\mathsf{d}} \sigma'_i, \Gamma_3, x : \oplus_{i\in H\cup K}\overline{a}_i.\rho_i \dashv^{\mathsf{d}} \Sigma_{j\in J'}\, a_j.\sigma'_j\big) = \mathcal{D}'_i :: \Gamma_3, x : \rho \dashv^{\mathsf{d}} \sigma' \rhd^{\mathsf{inf}} h_i : \rho_i \dashv^{\mathsf{d}} \sigma'$.

We then return

$$\dfrac{\begin{array}{cc}\boxed{\mathcal{D}'_i} & \boxed{\mathcal{D}'_i} \\ \Gamma_3, x : \rho \dashv^{\mathsf{d}} \sigma' \rhd^{\mathsf{inf}} h_i : \rho_i \dashv^{\mathsf{d}} \sigma' & \Gamma_3, x : \rho \dashv^{\mathsf{d}} \sigma' \rhd^{\mathsf{inf}} h_i : \rho_i \dashv^{\mathsf{d}} \sigma'_i \\ \vdots \quad (\forall i \in H \cup (H' \cap K)) & \vdots \quad (\forall i \in K' \cap K)\end{array}}{\Gamma_3 \rhd \bigvee_{i\in H\cup (H'\cap K)}\langle a_i, \varepsilon\rangle.h_i) \vee (\bigvee_{i\in K'\cap K}\langle a_i, \overline{a}_i\rangle.h_i : \rho \dashv^{\mathsf{d}} \sigma'} \; (\text{CPL}\oplus\text{-}\Sigma)$$

It is possible to define an operation of orchestrator composition, such that orchestrator of the result of the previous algorithm can be obtained by means of such operator.

Definition D.2 *(Orchestrator composition).* The composition $f \bullet g$ of two orchestrators f and g is defined by the rules in Fig. D.9, which we assume to be applied according to the priority respecting the order they are listed in.

The well-foundness of $f \bullet g$ can be shown from the regularity of f and g. The orchestrator $f \bullet g$ is actually the one returned by the procedure **P** that can be proved to terminate.

We say that $f \bullet g$ is defined whenever $f \bullet g \neq \bot$.

In the following Proposition we show the procedure **P** to terminate. Notice that we need to assume the orchestrators f and g in the first and second argument of **P** to be sound. In fact, f could be an orchestrator communicating only with the server and g one communicating only with the client. In that case, the procedure **P** would unsuccessfully try to produce an orchestrator made just of actions which are the result of two annihilating actions of the form $\langle \varepsilon, \overline{a}\rangle$ and $\langle \overline{a}, \varepsilon\rangle$, a sort of empty orchestrator.

This argument can be better understood by means of a very simple example. Note that it is possible to have two derivations for the judgements

$$\rhd^{\mathsf{inf}} \mathsf{rec}\, x.\langle \varepsilon, \overline{a}\rangle.x : b \dashv^{\mathsf{d}} \mathsf{rec}\, x.a.x \;\text{ and }\; \rhd^{\mathsf{inf}} \mathsf{rec}\, x.\langle a, \varepsilon\rangle.x : \mathsf{rec}\, x.\overline{a}.x \dashv^{\mathsf{d}} c$$

where $\mathsf{rec}\, x.\langle \varepsilon, \overline{a}\rangle.x$ is clearly unsound. The procedure **P** tries to build an orchestrator h and a derivation for

$$\rhd^{\mathsf{inf}} h : b \dashv^{\mathsf{d}} c$$

that it actually cannot do. In fact, **P** indefinitely keeps applying clause (CPLΣ-R)-(CPL⊕-Σ), which simply *annihilates* the actions $\langle \varepsilon, \overline{a}\rangle$ and $\langle a, \varepsilon\rangle$, without managing to produce an orchestration action of h (which cannot be produced out of the given orchestrators).

$$
\begin{array}{rcl}
\mathbf{1} \bullet g & = & \mathbf{1} \\
(\langle \overline{a}_p, \varepsilon \rangle.f') \bullet g & = & \langle \overline{a}_p, \varepsilon \rangle.(f' \bullet g) \\
f \bullet (\langle \varepsilon, \overline{a}_p \rangle.g') & = & (\langle \varepsilon, \overline{a}_p \rangle.(f \bullet g') \\
f \bullet (\bigvee_{i\in I} \langle \varepsilon, a_i \rangle.g_i) & = & \bigvee_{i\in I} \langle \varepsilon, a_i \rangle.(f \bullet g_i) \\
(\bigvee_{i\in I} \langle a_i, \varepsilon \rangle.f_i) \bullet g & = & \bigvee_{i\in I} \langle a_i, \varepsilon \rangle.(f_i \bullet g) \\
(\langle \varepsilon, \overline{a}_p \rangle.f') \bullet ((\bigvee_{h\in H} \langle a_h, \varepsilon \rangle.g_h) \vee (\bigvee_{k\in K} \langle a_k, \overline{a}_k \rangle.g_k)) & & \\
\qquad = f' \bullet g_p & & (p \in H) \\
(\langle \varepsilon, \overline{a}_p \rangle.f') \bullet ((\bigvee_{h\in H} \langle a_h, \varepsilon \rangle.g_h) \vee (\bigvee_{k\in K} \langle a_k, \overline{a}_k \rangle.g_k)) & & \\
\qquad = \langle \varepsilon, \overline{a}_p \rangle.f' \bullet g_p & & (p \in K) \\
((\bigvee_{i\in H} \langle \varepsilon, a_i \rangle.f_i) \vee (\bigvee_{k\in K} \langle \overline{a}_k, a_k \rangle.f_k)) \bullet (\langle \overline{a}_p, \varepsilon \rangle.g') & & \\
\qquad = f_p \bullet g' & & (p \in H) \\
((\bigvee_{i\in H} \langle \varepsilon, a_i \rangle.f_i) \vee (\bigvee_{k\in K} \langle \overline{a}_k, a_k \rangle.f_k)) \bullet (\langle \overline{a}_p, \varepsilon \rangle.g') & & \\
\qquad = \langle \overline{a}_p, \varepsilon \rangle(f_p \bullet g') & & (p \in K) \\
((\bigvee_{i\in H} \langle \varepsilon, a_i \rangle.f_i) \vee (\bigvee_{k\in K} \langle \overline{a}_k, a_k \rangle.f_k)) \bullet ((\bigvee_{i\in H'} \langle \varepsilon, a_i \rangle.g_i) \vee (\bigvee_{i\in K'} \langle \overline{a}_i, a_i \rangle.g_i)) = & & \\
\qquad \bigvee_{i\in H'} \langle \varepsilon, a_i \rangle.(f \bullet g_i) \vee \bigvee_{i\in H\cap K'} \langle \varepsilon, a_i \rangle.(f_i \bullet g_i) \vee \bigvee_{i\in K\cap K'} \langle \overline{a}_i, a_i \rangle.(f_i \bullet g_i) & & \\
\qquad (H \cup K \supseteq K') & & \\
((\bigvee_{h\in H} \langle a_h, \varepsilon \rangle.f_h) \vee (\bigvee_{k\in K} \langle a_k, \overline{a}_k \rangle.f_k)) \bullet ((\bigvee_{h\in H'} \langle a_h, \varepsilon \rangle.g_h) \vee (\bigvee_{k\in K'} \langle a_k, \overline{a}_k \rangle.g_k) = & & \\
\qquad \bigvee_{i\in H} \langle a_i, \varepsilon \rangle.(f_i \bullet g) \vee \bigvee_{i\in (H'\cap K)} \langle a_i, \varepsilon \rangle.(f_i \bullet g_i) \vee \bigvee_{i\in (K'\cap K)} \langle a_i, \overline{a}_i \rangle.(f_i \bullet g_i) & & \\
\qquad (H' \cup K' \supseteq K) & & \\
f \bullet g & = & \bot
\end{array}
$$

Fig. D.9. Composition of orchestrators.

Proposition D.3. *Assume* $\mathcal{D}_1 :: \rhd^{\mathsf{inf}} f : \rho \dashv^{\mathsf{d}} \sigma$ *and* $\mathcal{D}_2 :: \rhd^{\mathsf{inf}} g : \overline{\sigma} \dashv^{\mathsf{d}} \sigma'$, *where both* f *and* g *are sound orchestrators. Then*

1. *the computation of* $\mathbf{P}\left(\mathcal{D}_1, \mathcal{D}_2, \emptyset\right)$ *terminates;*
2. $f \bullet g$ *is defined;*
3. $\mathbf{P}\left(\mathcal{D}_1, \mathcal{D}_2, \emptyset\right) = \mathcal{D} :: \rhd^{\mathsf{inf}} f \bullet g : \rho \dashv^{\mathsf{d}} \sigma'$.

Proof.

1. The algorithm **P** is a derivation reconstruction algorithm that tries and build a derivation for the judgement $\rhd^{\mathsf{inf}} f \bullet g : \rho \dashv^{\mathsf{d}} \sigma'$ in a bottom-up way, driven by the two derivations $\mathcal{D}_1$ and $\mathcal{D}_2$, and by the third argument (the environment). The proof of its termination is basically rooted in the same ideas as the termination proof for the reconstruction algorithm **Prove** (Lemma A.10) used in the completeness proof for the formal system $\rhd$. Hence in the following we avoid detailing notions similar to those used there.
   We call the clauses belonging to the following set *invariant*:

$$
\mathcal{I}nv = \left\{ \begin{array}{c} (\textsc{Hyp}) \cdot (\textsc{Hyp}),\ (\textsc{Hyp}) \cdot *,\ * \cdot (\textsc{Hyp}),\ (\textsc{Cpl}\Sigma\text{-R}) \cdot (\textsc{Cpl}\oplus\text{-L}), \\ (\textsc{Cpl}\Sigma\text{-R}) \cdot (\textsc{Cpl}\oplus\text{-}\Sigma) \cdot (p \in H),\ (\textsc{Cpl}\oplus\text{-R}) \cdot (\textsc{Cpl}\Sigma\text{-L}) \end{array} \right\}
$$

   Also, given a call $\mathbf{P}\left(\mathcal{D} :: \Gamma_1 \rhd^{\mathsf{inf}} f : \rho \dashv^{\mathsf{d}} \sigma,\ \mathcal{D}' :: \Gamma_2 \rhd^{\mathsf{inf}} g : \overline{\sigma} \dashv^{\mathsf{d}} \sigma',\ \Gamma_3\right)$ of the procedure **P**, we refer to ρ as *the client*, to σ' as *the server*, to σ as *the intermediate server* and to $\overline{\sigma}$ as *the intermediate client* of the call.
   We first observe that for any recursive call corresponding to an application of a clause not in $\mathcal{I}nv$, both the client and the server in the call are subterms of the client and the server considered in the calling procedure. By the regularity of the trees represented by session contracts, it follows that the clauses (Ax)-$*$ or **Init** will apply. In fact, if f' and g' are the client and server of a call of **P** and if f'' and g'' are the client and server of the resulting recursive call, then the regular trees corresponding to f'' and g'' are, respectively, subtrees of the trees corresponding to f' and g'. This implies that the clients and servers of any call are subtrees of the client and server of the initial call. So if the sequence of recursive calls does not get to an application of clause (Ax)-$*$, eventually it will get to an application of clause **Init**, since the environment necessarily contains $\rho \dashv^{\mathsf{d}} \sigma$, where ρ and σ are the client and server of the initial call. This immediately implies that the derivation reconstruction does terminate.
   It remains to be proved that the initial call of the procedure **P** cannot produce an infinite sequence of recursive calls due only to the invariant clauses in $\mathcal{I}nv$. Towards a contradiction, assume that such a sequence exists. We distinguish two cases.
   The first is when the calls corresponding to $(\textsc{Cpl}\Sigma\text{-R}) \cdot (\textsc{Cpl}\oplus\text{-L})$, $(\textsc{Cpl}\Sigma\text{-R}) \cdot (\textsc{Cpl}\oplus\text{-}\Sigma) \cdot (p \in H)$ and $(\textsc{Cpl}\oplus\text{-R}) \cdot (\textsc{Cpl}\Sigma\text{-L})$ are finite. This means that from a certain point onwards, the sequence is made just of calls corresponding to $(\textsc{Hyp}) \cdot (\textsc{Hyp})$,

$$
\begin{array}{lrcl}
1. & 1 \bullet \boldsymbol{\mu} & = & 1 \\
2. & (\langle \overline{a}, \varepsilon \rangle . \boldsymbol{\mu_1}) \bullet \boldsymbol{\mu_2} & = & \langle \overline{a}, \varepsilon \rangle . (\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2}) \\
3. & (\langle \overline{a}, a \rangle . \boldsymbol{\mu_1}) \bullet \langle \overline{a}, \epsilon \rangle . \boldsymbol{\mu_2} & = & \langle \overline{a}, \varepsilon \rangle . (\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2}) \\
4. & \boldsymbol{\mu_1} \bullet (\langle \varepsilon, \overline{a} \rangle . \boldsymbol{\mu_2}) & = & \langle \varepsilon, \overline{a} \rangle . (\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2}) \\
5. & (\langle \varepsilon, \overline{a} \rangle . \boldsymbol{\mu_1}) \bullet (\langle a, \varepsilon \rangle . \boldsymbol{\mu_2}) & = & \boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2} \\
6. & (\langle \varepsilon, a \rangle . \boldsymbol{\mu_1}) \bullet (\langle \overline{a}, \varepsilon \rangle . \boldsymbol{\mu_2}) & = & \boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2} \\
7. & \boldsymbol{\mu_1} \bullet (\langle \varepsilon, a \rangle . \boldsymbol{\mu_2}) & = & \langle \varepsilon, a \rangle . (\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2}) \\
8. & (\langle \varepsilon, a \rangle . \boldsymbol{\mu_1}) \bullet (\langle \overline{a}, a \rangle . \boldsymbol{\mu_2}) & = & \langle \varepsilon, a \rangle . (\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2}) \\
9. & (\langle a, \varepsilon \rangle . \boldsymbol{\mu_1}) \bullet \boldsymbol{\mu_2} & = & \langle a, \varepsilon \rangle . (\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2}) \\
10. & (\langle a, \overline{a} \rangle . \boldsymbol{\mu_1}) \bullet (\langle a, \varepsilon \rangle . \boldsymbol{\mu_2}) & = & \langle a, \varepsilon \rangle . (\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2}) \\
11. & \boldsymbol{\mu_1} \bullet (\langle a, \varepsilon \rangle . \boldsymbol{\mu_2}) & = & \langle a, \varepsilon \rangle . (\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2}) \\
12. & (\langle a, \overline{a} \rangle . \boldsymbol{\mu_1}) \bullet (\langle a, \overline{a} \rangle . \boldsymbol{\mu_2}) & = & \langle a, \overline{a} \rangle . (\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2}) \\
13. & (\langle \overline{a}, a \rangle . \boldsymbol{\mu_1}) \bullet (\langle \overline{a}, a \rangle . \boldsymbol{\mu_2}) & = & \langle \overline{a}, a \rangle . (\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2}) \\
14. & \boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2} & = & \bot
\end{array}
$$

Fig. D.10. Composition of traces.

(Hyp) $\cdot *$ and $* \cdot$ (Hyp). This is however impossible, since the syntax of session contracts impose that the body σ of a term $\mathrm{rec}\, x.\sigma$ is not a variable.
In case the calls corresponding to (CplΣ-R) $\cdot$ (Cpl$\oplus$-L), (CplΣ-R) $\cdot$ (Cpl$\oplus$-Σ) $\cdot$ ($p \in H$) and (Cpl$\oplus$-R) $\cdot$ (CplΣ-L) are infinite, we get a contradiction with the assumption that f and g are sound. In fact we could easily build an infinite trace of f or g which is not sound, consisting only of orchestration actions $\langle \varepsilon, \overline{a} \rangle$ and $\langle \varepsilon, a \rangle$ (in case of f) or orchestration actions $\langle \overline{a}, \varepsilon \rangle$ and $\langle a, \varepsilon \rangle$ (in case of g).

2. and 3. Easy by inspection of the clauses of the procedure **P**. □

We will now prove that if $\rhd^{\mathsf{inf}} f : \rho \dashv^{\mathsf{d}} \sigma$ and $\rhd^{\mathsf{inf}} g : \overline{\sigma} \dashv^{\mathsf{d}} \sigma'$ and f and g are respectful, then so is $f \bullet g$. Since respectfulness is a property of traces, we have to check the respectfulness of the traces of $f \bullet g$. We observe that a trace of $f \bullet g$ can be seen as the composition of a trace of f and a trace of g. Therefore we extend the composition operator to traces.

Definition D.4 *(Traces composition).* Composition of traces is defined through the rules in Fig. D.10; these are assumed to be applied respecting the priority of their order.

We say that $\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2}$ is *defined* whenever $\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2} \neq \bot$.

Lemma D.5. *Let f and g be such that $f \bullet g$ is defined. Then*

$$\boldsymbol{\mu} \in \mathsf{Tr}(f \bullet g) \implies \exists\, \boldsymbol{\mu_1} \in \mathsf{Tr}(f)\ \exists\, \boldsymbol{\mu_2} \in \mathsf{Tr}(g)\ [\, \boldsymbol{\mu} = \boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2} \,]$$

Proof. Easy, by definition of traces and by checking the rules of the definitions of orchestrators and traces composition (Definition D.2 and D.4). □

For the sake of readability we write $\boldsymbol{\mu}$ instead of $\tilde{\emptyset}\boldsymbol{\mu}$ when no ambiguity can arise. In the following the subsequence of the first n elements of $\boldsymbol{\mu}$ will be denoted by $(\boldsymbol{\mu})_n$.
We will prove that the composition of two respectful traces is respectful itself. Since respectfulness is the conjunction of other properties, we split the proof into some lemmas about these properties.

Lemma D.6 *(Soundness preservation). Let $\boldsymbol{\mu_1}$ and $\boldsymbol{\mu_2}$ be such that $\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2}$ is defined.*

1. *For any $a \in \mathcal{N}$ and $n \geq 0$, there exists $k_1, k_2, h_1, h_2 \geq 0$ such that*

$$ {}_a|(\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2})_n| \;=\; {}_a|(\boldsymbol{\mu_1})_{k_1}| + {}_a|(\boldsymbol{\mu_2})_{k_2}| $$

and

$$ |(\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2})_n|_a \;=\; |(\boldsymbol{\mu_1})_{h_1}|_a + |(\boldsymbol{\mu_2})_{h_2}|_a $$

2. *$\boldsymbol{\mu_1}$ and $\boldsymbol{\mu_2}$ are sound $\implies$ $\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2}$ is sound.*

Proof.

1. We take into account just the first property, the second one can be proved in a similar way. Any element in $(\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2})_n$ corresponds to an application of one of the rules for merging traces in Definition D.4. Given n, let k be number of merging rules necessary to get $(\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2})_n$. We get the thesis by observing that when we merge finite traces, all the rules of Definition D.4 respect the property.
2. By contraposition, assume $\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2}$ is not sound. Then, by definition of sound sequence (Definition 4.3), there exists $a \in \mathcal{N}$ and $n \geq 0$ such that ${}_a|(\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2})_n| < 0$ or $|(\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2})_n|_a < 0$. By (1) and definition of sound sequence, this would imply either $\boldsymbol{\mu_1}$ or $\boldsymbol{\mu_2}$ not to be sound. □

Lemma D.7 *(Client-respectfulness preservation). Let $\boldsymbol{\mu_1}$ and $\boldsymbol{\mu_2}$ be such that $\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2}$ is defined. Then*

1. ${}_a\downarrow(\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2}) = {}_a\downarrow\boldsymbol{\mu_1} \bullet {}_a\downarrow\boldsymbol{\mu_2}$.
2. *$\boldsymbol{\mu_1}$ and $\boldsymbol{\mu_2}$ client-respectful $\implies$ $\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2}$ client-respectful.*

Proof.

1. The property can be coinductively proved in case $\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2}$ is infinite, and inductively in the finite case, by checking that ${}_a\downarrow\boldsymbol{\mu_1} \bullet {}_a\downarrow\boldsymbol{\mu_2} = {}_a\downarrow(\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2})$ implies that, for any rule $\boldsymbol{\mu'} \bullet \boldsymbol{\mu''} = \boldsymbol{\mu'''}$ in Definition D.4, but the first and the last one, ${}_a\downarrow\boldsymbol{\mu'} \bullet {}_a\downarrow\boldsymbol{\mu''} = {}_a\downarrow\boldsymbol{\mu'''}$ holds. We show this for some rules. All others can be treated similarly.
$(\langle \varepsilon, \overline{a} \rangle.\boldsymbol{\mu_1}) \bullet (\langle a, \varepsilon \rangle.\boldsymbol{\mu_2}) = \boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2}$:

$$\begin{aligned} {}_a\downarrow(\langle \varepsilon, \overline{a} \rangle.\boldsymbol{\mu_1}) \bullet {}_a\downarrow(\langle a, \varepsilon \rangle.\boldsymbol{\mu_2}) &= (\langle \varepsilon, \overline{a} \rangle.{}_a\downarrow\boldsymbol{\mu_1}) \bullet (\langle a, \varepsilon \rangle.{}_a\downarrow\boldsymbol{\mu_2}) \\ &= {}_a\downarrow\boldsymbol{\mu_1} \bullet {}_a\downarrow\boldsymbol{\mu_2} \\ &= {}_a\downarrow(\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2}) \end{aligned}$$

$(\langle a, \overline{a} \rangle.\boldsymbol{\mu_1}) \bullet (\langle a, \varepsilon \rangle.\boldsymbol{\mu_2}) = \langle a, \varepsilon \rangle.(\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2})$:

$$\begin{aligned} {}_a\downarrow(\langle a, \overline{a} \rangle.\boldsymbol{\mu_1}) \bullet {}_a\downarrow(\langle a, \varepsilon \rangle.\boldsymbol{\mu_2}) &= (\langle a, \overline{a} \rangle.{}_a\downarrow\boldsymbol{\mu_1}) \bullet (\langle a, \varepsilon \rangle.{}_a\downarrow\boldsymbol{\mu_2}) \\ &= \langle a, \varepsilon \rangle.({}_a\downarrow\boldsymbol{\mu_1} \bullet {}_a\downarrow\boldsymbol{\mu_2}) \\ &= {}_a\downarrow(\langle a, \varepsilon \rangle.(\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2})) \end{aligned}$$

$\boldsymbol{\mu_1} \bullet (\langle \varepsilon, a \rangle.\boldsymbol{\mu_2}) = \langle \varepsilon, a \rangle.(\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2})$:

$$\begin{aligned} {}_a\downarrow\boldsymbol{\mu_1} \bullet {}_a\downarrow(\langle \varepsilon, a \rangle.\boldsymbol{\mu_2}) &= {}_a\downarrow\boldsymbol{\mu_1} \bullet {}_a\downarrow\boldsymbol{\mu_2} \\ &= {}_a\downarrow(\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2}) \end{aligned}$$

2. By contraposition, assume that $\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2}$ is not client respectful. This means that there exists $a \in \mathcal{N}$ such that, in case ${}_a\downarrow(\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2})$ is finite, ${}_a|\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2}| \neq 0$, whereas, in case it is infinite, we have it is definitely-$\langle a, \varepsilon \rangle$.
In the first case, by Lemma D.6(1) we have that ${}_a|\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2}| = {}_a|\boldsymbol{\mu_1}| + {}_a|\boldsymbol{\mu_2}| \neq 0$. Since one of the two addenda must be different from 0, we get that either $\boldsymbol{\mu_1}$ or $\boldsymbol{\mu_2}$ is not client-respectful.
In the second case, by checking the rules of Definition D.4 using the property (1), we notice that the only way of getting ${}_a\downarrow(\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2})$ definitely-$\langle a, \varepsilon \rangle$ is when either ${}_a\downarrow\boldsymbol{\mu_1}$ or ${}_a\downarrow\boldsymbol{\mu_2}$ is definitely-$\langle a, \varepsilon \rangle$. □

Remark D.8. Notice that the non-definitely-inputted-ness is not necessarily preserved by $\bullet$. In fact both $f = \mathsf{rec}\, x.\langle \varepsilon, a \rangle.\langle \varepsilon, \overline{a} \rangle.\langle \varepsilon, a \rangle.x$ and $g = \mathsf{rec}\, x.\langle \overline{a}, a \rangle.\langle a, \varepsilon \rangle.\langle \overline{a}, \varepsilon \rangle.x$ are non-definitely inputted orchestrators, but

$$f \bullet g = \mathsf{rec}\, x.\langle \varepsilon, a \rangle.x$$

is definitely inputted. This is due to orchestration actions in f and g that are 'annihilated' in $f \bullet g$. This sort of behaviour is however made possible just by the unsoundness of f and g.

Lemma D.9 *(Non-definitely server inputted-ness preservation). Let $\boldsymbol{\mu_1}$ and $\boldsymbol{\mu_2}$ be such that $\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2}$ is defined.*

$\boldsymbol{\mu_1}$ and $\boldsymbol{\mu_2}$ are sound and non-definitely server inputted
implies
$\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2}$ is non-definitely server inputted.

Proof. By contraposition, assume, by definition of non-definitely server-inputted sequence (Definition 4.3(5), that $\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2}$ is definitely-$\{ \langle \varepsilon, a \rangle \mid a \in \mathcal{N} \}$.

Assume $\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2}$ to be definitely-$\{ \langle \varepsilon, a \rangle \mid a \in \mathcal{N} \}$ starting from its element k. In order to obtain a sequence out of $\boldsymbol{\mu_1}$ and $\boldsymbol{\mu_2}$ made only of elements in $\{ \langle \varepsilon, a \rangle \mid a \in \mathcal{N} \}$ from the element k onwards, only rules 5, 6, 7, and 8 of Definition D.4 can be used after producing the element k. We now distinguish three different cases.

- In case $\boldsymbol{\mu_1} \bullet \boldsymbol{\mu_2}$ is built just out of rules 7 and 8, then it follows that either $\boldsymbol{\mu_1}$ or $\boldsymbol{\mu_2}$ is definitely server inputted.
- In case rules 5 and 6 are used finitely many times after the production of the element k, the previous argument applies by taking into account the element $h \geq k$ of the sequence produced before the last application of rule 5 or 6.
- If rules 5 and 6 are applied infinitely many times, one of the two is applied infinitely many times. If it is rule 5, we can infer $\boldsymbol{\mu_1}$ not to be a sound sequence: in fact an infinite number of messages are sent to the server but only a finite number can be received from the client. If it is rule 6, we can infer that $\boldsymbol{\mu_2}$ is not a sound sequence: in fact an infinite number of messages are sent to the client but only a finite number can be received from the server. □

Corollary D.10 *(Respectfulness preservation). Let f and g be such that $f \bullet g$ is defined. Then*

$$f \text{ and } g \text{ respectful} \implies f \bullet g \text{ respectful.}$$

Proof. Easy from Lemma D.5, D.6(2), D.7(2) and D.9. □

Corollary D.11 *(Proposition 5.4(2)).* $\rho \dashv\!\shortmid \sigma \ \&\ \overline{\sigma} \dashv\!\shortmid \sigma' \implies \rho \dashv\!\shortmid \sigma'$.

Proof. Let $\rho \dashv\!\shortmid \sigma$ and $\overline{\sigma} \dashv\!\shortmid \sigma'$. Then, by definition, there exist two respectful f and g such that $f : \rho \dashv\!\shortmid \sigma$ and $g : \overline{\sigma} \dashv\!\shortmid \sigma'$. We can safely assume that f and g are strict. So, by completeness of system $\rhd$ and the equivalence of systems $\rhd$ and $\rhd^{\mathsf{inf}}$, we deduce that $\rhd^{\mathsf{inf}} f : \rho \dashv\!\shortmid^{\mathsf{d}} \sigma$ and $\rhd^{\mathsf{inf}} g : \overline{\sigma} \dashv\!\shortmid^{\mathsf{d}} \sigma'$. The thesis now follows by Proposition D.3 and Corollary D.10. □

We now proceed with the proof of Proposition 5.4(3).

Definition D.12 *(Synchronous orchestrators).* An orchestrator is said to be *synchronous* if all its orchestration actions are.

Lemma D.13. $\rho \dashv \sigma \implies \exists\, f \text{ synchronous } [\, f : \rho \dashv\!\shortmid^{\mathsf{d}} \sigma \,]$.

Proof. Easy, since the relation $\dashv$ is characterised via system $\rhd$ where only synchronous actions are taken into account. □

Proposition D.14. *Let f be synchronous.*

1. *f is respectful.*
2. *For any g such that $f \bullet g$ is defined, $f \bullet g \leq g$.*

Proof.

1. Easy, by definition of respectful orchestrator.
2. Easy, by definition of orchestrators composition (Definition D.2). □

Corollary D.15 *(Proposition 5.4(3)).* $\rho \dashv \sigma \ \&\ g : \overline{\sigma} \dashv\!\shortmid \sigma' \implies g : \rho \dashv\!\shortmid \sigma'$.

Proof. $\rho \dashv \sigma$ implies, by Lemmas D.13 and D.14(1), that $f : \rho \dashv\!\shortmid \sigma$ from some respectful f. Being f respectful, it is sound a fortiori. So, from $f : \rho \dashv\!\shortmid \sigma$ and $g : \overline{\sigma} \dashv\!\shortmid \sigma'$ we get $f \bullet g : \rho \dashv\!\shortmid \sigma'$ by soundness and completeness of system $\rhd^{\mathsf{inf}}$ and by Proposition D.3. The thesis follows by Proposition D.14(2) and Lemma 3.15. □

Appendix E. Proofs of Theorems 6.11 and 6.12

We start with the proof of Theorem 6.11, i.e.

$$\rho \dashv^{\mathtt{skp}} \sigma \quad \Leftrightarrow \quad \rho \dashv\!\shortmid^{\mathtt{skp}}_{\mathtt{Orch}} \sigma$$

We will prove the thesis by relating two formal systems characterising $\dashv^{\mathtt{skp}}$ and $\dashv\!\shortmid^{\mathtt{skp}}_{\mathtt{Orch}}$, respectively.

We begin with $\dashv^{\mathtt{skp}}$, noticing that the relation of $\mathtt{skp}$-compliance can be equivalently defined as follows, where $\dashv^{\mathtt{s}}$ is the notion of standard compliance using the LTS of Definition 6.1.

Definition E.1.

1. Let the relation $\dashv^{\mathtt{s}} \subseteq \mathsf{SC} \times \mathsf{SC}$ be defined by:

$$\rho \dashv^{\mathtt{s}} \sigma \ \triangleq\ \forall \boldsymbol{\nu} \in \mathbf{SkipAct}^*,\ \rho',\ \sigma'\, [\, \rho \parallel \sigma \stackrel{\boldsymbol{\nu}}{\Longrightarrow} \rho' \parallel \sigma' \not\rightarrow \implies \rho' = \mathbf{1}\,].$$

2. $\rho \dashv^{\mathtt{skp}} \sigma \triangleq \rho \dashv^{\mathtt{s}} \sigma$ and all infinite traces in $\mathsf{skTr}(\rho \parallel \sigma)$ are non-definitely-$\mathtt{skp}$.

$$(\text{Ax-s}):\ \frac{}{\Gamma \rhd^{s} \mathbf{1} \dashv^{s} \sigma} \qquad (\text{Hyp-s}):\ \frac{}{\Gamma, \rho \dashv^{s} \sigma \rhd^{s} \rho \dashv^{s} \sigma}$$

$$(+.\oplus\text{-s}):\ \frac{\Gamma' \rhd^{s} \Sigma_{k\in K}\, a_k.\rho_k \dashv^{s} \sigma_i \quad (\forall i \in I\backslash K) \qquad \Gamma'' \rhd^{s} \rho_j \dashv^{s} \sigma_j \quad (\forall j \in K \cap I)}{\Gamma \rhd^{s} \Sigma_{k\in K}\, a_k.\rho_k \dashv^{s} \oplus_{i\in I}\, \overline{a}_i.\sigma_i}$$
where $\Gamma' = \Gamma,\ \Sigma_{k\in K}\, a_k.\rho_k \dashv^{s} \oplus_{i\in I}\, \overline{a}_i.\sigma_i$

$$(\oplus.\oplus\text{-s}):\ \frac{\Gamma' \rhd^{s} \oplus_{k\in K}\, \overline{a}_k.\rho_k \dashv^{s} \sigma_i \quad (\forall i \in I)}{\Gamma \rhd^{s} \oplus_{k\in K}\, \overline{a}_k.\rho_k \dashv^{s} \oplus_{i\in I}\, \overline{b}_i.\sigma_i}$$
where $\Gamma' = \Gamma,\ \oplus_{k\in K}\, \overline{a}_k.\rho_k \dashv^{s} \oplus_{i\in I}\, \overline{b}_i.\sigma_i$

$$(\oplus.+\text{-s}):\ \frac{\Gamma' \rhd^{s} \rho_k \dashv^{s} \sigma_k \quad (\forall k \in K)\ (K \subseteq I)}{\Gamma \rhd^{s} \oplus_{k\in K}\, \overline{a}_k.\rho_k \dashv^{s} \Sigma_{i\in I}\, a_i.\sigma_i}$$
where $\Gamma' = \Gamma,\ \oplus_{k\in K}\, \overline{a}_k.\rho_k \dashv^{s} \Sigma_{i\in I}\, a_i.\sigma_i$

Fig. E.11. The formal system $\rhd^{s}$.

Fact E.2. *Definitions 6.3 and E.1(2) are equivalent.*

By means of proofs precisely mimicking those in Section 3 for the corresponding statements for system $\rhd$ (without taking into account the presence of orchestrators), it is possible to prove the following.

Proposition E.3. *Let $\rhd^{s}$ be the formal system of Fig. E.11.*

1. *System $\rhd^{s}$ is sound and complete with respect to the relation $\dashv^{s}$:*
$$\rhd^{s} f : \rho \dashv^{s} \sigma \quad \Leftrightarrow \quad f : \rho \dashv^{s} \sigma$$
2. *Proof search always terminate for system $\rhd^{s}$.*

We provide now a restricted version of Definition 2.10. We call a pair $\rho \parallel_f \sigma$ a `skipOrch`-system when f is a skipping orchestrator.

Definition E.4 *(Operational semantics of* `skipOrch` *systems).* Given $\rho, \sigma \in \mathsf{SC}$ and $f \in \mathsf{SkpOrch}$. The operational semantics of the `skipOrch` orchestrated system $\rho \parallel_f \sigma$ is defined as follows (where $a \in \mathcal{N}$):

$$\frac{\rho \xrightarrow{\tau} \rho'}{\rho \parallel_f \sigma \xrightarrow{\tau}_{-} \rho' \parallel_f \sigma} \qquad \frac{\sigma \xrightarrow{\tau} \sigma'}{\rho \parallel_f \sigma \xrightarrow{\tau}_{-} \rho \parallel_f \sigma'}$$

$$\frac{\rho \xrightarrow{\alpha} \rho' \quad f \overset{\langle \overline{\alpha}, \alpha \rangle}{\longmapsto} f' \quad \sigma \xrightarrow{\overline{\alpha}} \sigma'}{\rho \parallel_f \sigma \xrightarrow{\langle \overline{\alpha}, \alpha \rangle}_{-} \rho' \parallel_{f'} \sigma'} \qquad \frac{\rho \not\downarrow a \quad f \overset{\langle \varepsilon, a \rangle}{\longmapsto} f' \quad \sigma \xrightarrow{\overline{a}} \sigma'}{\rho \parallel_f \sigma \xrightarrow{\langle \varepsilon, a \rangle}_{-} \rho \parallel_{f'} \sigma'}$$

We write $\xRightarrow{\mu}_{-}$ for $\xrightarrow{\tau}_{-}{}^{*} \circ \xrightarrow{\mu}_{-} \circ \xrightarrow{\tau}_{-}{}^{*}$, and $\xRightarrow{\boldsymbol{\mu}}_{-}$ for $\xRightarrow{\mu_1}_{-} \circ \cdots \circ \xRightarrow{\mu_n}_{-}$ (resp. $\xRightarrow{\mu_1}_{-} \circ \xRightarrow{\mu_2}_{-} \circ \cdots$) if the sequence $\boldsymbol{\mu}$ is finite (resp. infinite).

The notation $\rho \parallel_f \sigma \longrightarrow_{-}$ will be used for either $\rho \parallel_f \sigma \xrightarrow{\tau}_{-}$ or $\exists \mu\ [\, \rho \parallel_f \sigma \xrightarrow{\mu}_{-}\,]$.

We now define:

Definition E.5 *(Restricted orchestrated compliance).* Let $\rho, \sigma \in \mathsf{SC}$ and $f \in \mathsf{SkpOrch}$. We define:

i) $f : \rho \dashv\!\underline{\ }^{\mathsf{d}}\, \sigma$ if for any $\boldsymbol{\mu} \in \mathbf{SkipAct}$, $f' \in \mathsf{SkpOrch}$ and $\rho', \sigma' \in \mathsf{SC}$, the following holds:
$$\rho \parallel_f \sigma \xRightarrow{\boldsymbol{\mu}}_{-} \rho' \parallel_{f'} \sigma' \not\longrightarrow_{-} \text{ implies } \rho' = \mathbf{1}.$$

ii) $\rho \dashv\!\underline{\ }^{\mathsf{d}}\, \sigma \triangleq \exists f\ [\, f : \rho \dashv\!\underline{\ }^{\mathsf{d}}\, \sigma\,]$.

$$(\text{Ax}):\ \frac{}{\Gamma \rhd^{\text{inf}}_{\text{sO}} 1 : \mathbf{1} \dashv^{\text{d}}_{-} \sigma} \qquad (\text{Hyp}):\ \frac{}{\Gamma, x{:}\rho \dashv^{\text{d}}_{-} \sigma \rhd^{\text{inf}}_{\text{sO}} x : \rho \dashv^{\text{d}}_{-} \sigma}$$

$$(\text{Cpl}\Sigma\text{-}\oplus)^{-}:\ \frac{\Gamma' \rhd^{\text{inf}}_{\text{sO}} f_i : \Sigma_{k\in K}\, a_k.\rho_k \dashv^{\text{d}}_{-} \sigma_i \quad (\forall i \in (K \setminus I)) \qquad \Gamma' \rhd^{\text{inf}}_{\text{sO}} f_j : \rho_j \dashv^{\text{d}}_{-} \sigma_j \quad (\forall j \in (K \cap I))}{\Gamma \rhd^{\text{inf}}_{\text{sO}} f : \Sigma_{k\in K}\, a_k.\rho_k \dashv^{\text{d}}_{-} \oplus_{i\in I} \overline{a}_i.\sigma_i}$$

where $\Gamma' = \Gamma,\ x{:}\Sigma_{i\in I}\, \overline{a}_i.\rho_i \dashv^{\text{d}}_{-} \oplus_{j\in J} a_j.\sigma_j$
and $f = \text{rec}\, x.((\bigvee_{i\in(K\setminus I)} \langle \varepsilon, a_i\rangle.f_i) \vee (\bigvee_{j\in(K\cap I)} \langle \overline{a}_j, a_j\rangle.f_j))$

$$(\text{Cpl}\oplus\text{-R}):\ \frac{\Gamma' \rhd^{\text{inf}}_{\text{sO}} f_j : \oplus_{i\in I}\, \overline{a}_i.\rho_i \dashv^{\text{d}}_{-} \sigma_j \quad (\forall j \in J)}{\Gamma \rhd^{\text{inf}}_{\text{sO}} \text{rec}\, x.\bigvee_{j\in J} \langle \varepsilon, b_j\rangle.f_j : \oplus_{i\in I}\, \overline{a}_i.\rho_i \dashv^{\text{d}}_{-} \oplus_{j\in J}\, \overline{b}_j.\sigma_i}$$

where $\Gamma' = \Gamma,\ x{:}\oplus_{i\in I}\, \overline{a}_i.\rho_i \dashv^{\text{d}}_{-} \oplus_{j\in J}\, \overline{b}_j.\sigma_j$

$$(\text{Cpl}\oplus\text{-}\Sigma):\ \frac{\Gamma' \rhd^{\text{inf}}_{\text{sO}} f_i : \rho_i \dashv^{\text{d}}_{-} \sigma_i \quad (\forall i \in I)}{\Gamma \rhd^{\text{inf}}_{\text{sO}} \text{rec}\, x.\bigvee_{i\in i} \langle a_i, \overline{a}_i\rangle.f_i : \oplus_{i\in I}\, \overline{a}_i.\rho_i \dashv^{\text{d}}_{-} \Sigma_{j\in J}\, a_j.\sigma_j}\ (I \subseteq J)$$

where $\Gamma' = \Gamma,\ x{:}\oplus_{i\in I}\, \overline{a}_i.\rho_i \dashv^{\text{d}}_{-} \Sigma_{j\in J}\, a_j.\sigma_j$

Fig. E.12. The inference system $\rhd^{\text{inf}}_{\text{sO}}$.

With the above definitions it is easy to prove the following:

Fact E.6. *If* $f : \rho \dashv\!\vdash^{\text{d}}_{-} \sigma$ *then* $f \in (\rho \parallel \sigma)$-SkpOrch.

Now, by mimicking the proof of Section 3 it is possible to device a sound and complete system for the relation $\dashv\!\vdash^{\text{d}}_{-}$.

Proposition E.7. *Let* $\rhd^{\text{inf}}_{\text{sO}}$ *be the formal system of Fig. E.12*

1. *System* $\rhd^{\text{inf}}_{\text{sO}}$ *is sound and complete with respect to the relation* $\dashv\!\vdash^{\text{d}}_{-}$*:*
$$\rhd^{\text{inf}}_{\text{sO}} f : \rho \dashv^{\text{d}}_{-} \sigma \quad \Leftrightarrow \quad f : \rho \dashv\!\vdash^{\text{d}}_{-} \sigma\,.$$
2. *Proof search always terminate for system* $\rhd^{\text{inf}}_{\text{sO}}$.

System $\rhd^{\text{inf}}_{\text{sO}}$ is essentially the restriction to skipping orchestrators of System $\rhd^{\text{inf}}$. Notice that, whereas Rule $(\text{Cpl}\Sigma\text{-}\oplus)$ in $\rhd^{\text{inf}}$ does correspond to a whole set of rules (one for each possibility of partitioning J into the sets H and K), Rule $(\text{Cpl}\Sigma\text{-}\oplus)^{-}$ in $\rhd^{\text{inf}}_{\text{sO}}$ is one single rule.

Fact E.8. $\rhd^{\text{inf}}_{\text{sO}} f : \rho \dashv^{\text{d}}_{-} \sigma$ *implies* $\rhd^{\text{inf}} f : \rho \dashv^{\text{d}} \sigma$.

By a simple inspection of the rules of systems $\rhd^{\text{s}}$ and $\rhd^{\text{inf}}_{\text{sO}}$, it is easy to get the following:

Fact E.9. *There exists a one-to-one correspondence between derivations in* $\rhd^{\text{s}}$ *and derivations in* $\rhd^{\text{inf}}_{\text{sO}}$.

Systems $\rhd^{\text{s}}$ and $\rhd^{\text{inf}}_{\text{sO}}$ are related as follows:

Lemma E.10. $\rhd^{\text{s}} \rho \dashv^{\text{s}} \sigma \ \Leftrightarrow\ \exists f \in (\rho \parallel \sigma)$-SkpOrch $[\, \rhd^{\text{inf}}_{\text{sO}} f : \rho \dashv^{\text{d}}_{-} \sigma\,]$.

Proof. By the soundness and completeness property of $\rhd^{\text{s}}$ and $\rhd^{\text{inf}}_{\text{sO}}$ and Facts E.9 and E.6. □

When $\rhd^{\text{s}} \rho \dashv^{\text{s}} \sigma$, any reduction sequence out of $\rho \parallel \sigma$ for the LTS of Definition 6.1 corresponds to a reduction sequence out of $\rho \parallel_f \sigma$ for the LTS of orchestrated systems.

Lemma E.11. *Let* $\rhd^{\text{inf}}_{\text{sO}} f : \rho \dashv^{\text{d}}_{-} \sigma$*, then:* f *is ndsi if and only if all the infinite traces in* $\text{skTr}(\rho \parallel \sigma)$ *are non-definitely-*skp.

Proof. From $\rhd^{\text{inf}}_{\text{sO}} f : \rho \dashv^{\text{d}}_{-} \sigma$, by Lemma E.10 and Fact E.6 and by soundness and completeness, we get that both $f : \rho \dashv\!\vdash^{\text{d}}_{-} \sigma$ and $\rho \dashv^{\text{s}} \sigma$ hold, with $f \in (\rho \parallel \sigma)$-SkpOrch. It follows, by definition of the LTSs on which the relations $\dashv\!\vdash^{\text{d}}_{-}$ and $\dashv^{\text{s}}$ are

based, that any infinite reduction sequence out of $\rho \parallel \sigma$ corresponds to an infinite reduction sequence out of $\rho \parallel_f \sigma$ and vice versa. The thesis then follows from the fact that f is $\rho \cdot \sigma$-strict, which in turn is an immediate consequence of Fact E.8. □

We are now ready to prove the left-to-right part of Theorem 6.11.

Proposition E.12. $\rho \dashv^{\mathtt{skp}} \sigma \implies \rho \dashv\!\vdash^{\mathtt{skp}}_{\mathtt{Orch}} \sigma$.

Proof. From $\rho \dashv^{\mathtt{skp}} \sigma$ we get, by Definition E.1(2), $\rho \dashv^{\mathsf{s}} \sigma$ and that and all the infinite traces in $\mathsf{sk\Gamma r}(\rho \parallel \sigma)$ are non-definitely-$\mathtt{skp}$. By Proposition E.3(1) we get that $\rhd^{\mathsf{s}} \rho \dashv^{\mathsf{s}} \sigma$. By Lemma E.10 there exists $f \in (\rho \parallel \sigma)$-SkpOrch such that $\rhd^{\mathsf{inf}}_{\mathsf{sO}} f : \rho \dashv\!\vdash^{\mathsf{d}}_{\_} \sigma$. Moreover, f is ndsi by Lemma E.11. By Fact E.8 we get that $\rhd^{\mathsf{inf}} f : \rho \dashv^{\mathsf{d}} \sigma$ and hence, by completeness, $f : \rho \dashv\!\vdash^{\mathsf{d}} \sigma$. We conclude $\rho \dashv\!\vdash^{\mathtt{skp}}_{\mathtt{Orch}} \sigma$ by Fact 6.10(2). □

We proceed now towards the proof of the opposite direction of Proposition E.12.

Lemma E.13. $f : \rho \dashv\!\vdash^{\mathsf{d}}_{\_} \sigma \iff f : \rho \dashv\!\vdash^{\mathsf{d}} \sigma \ \&\ f \in (\rho \parallel \sigma)$-SkpOrch.

Proof. The *only if* part immediately follows from Fact E.8 and definition of the LTS on which $\dashv\!\vdash^{\mathsf{d}}_{\_}$ is based. For the *if* part it suffices to observe that, by the fact that f belongs to $(\rho \parallel \sigma)$-SkpOrch, any sequence of reductions out of $\rho \parallel_f \sigma$ is necessarily made out of $\rightarrow_{\_}$ reductions (Definition E.4). □

Proposition E.14. $\rho \dashv\!\vdash^{\mathtt{skp}}_{\mathtt{Orch}} \sigma \implies \rho \dashv^{\mathtt{skp}} \sigma$.

Proof. Let $f : \rho \dashv\!\vdash^{\mathtt{skp}}_{\mathtt{Orch}} \sigma$, then by Fact 6.10(1) and Lemma E.13 we get that $f : \rho \dashv\!\vdash^{\mathsf{d}}_{\_} \sigma$, $f \in (\rho \parallel \sigma)$-SkpOrch and f is ndsi. So, by completeness of System $\rhd^{\mathsf{inf}}_{\mathsf{sO}}$, we get that $\rhd^{\mathsf{inf}}_{\mathsf{sO}} f : \rho \dashv\!\vdash^{\mathsf{d}}_{\_} \sigma$, $f \in (\rho \parallel \sigma)$-SkpOrch and f is ndsi. Now by Lemma E.11 we get that all the infinite traces in $\mathsf{sk\Gamma r}(\rho \parallel \sigma)$ are non-definitely-$\mathtt{skp}$, whereas from Lemma E.10 we get $\rhd^{\mathsf{s}} \rho \dashv^{\mathsf{s}} \sigma$. By the soundness of System $\rhd^{\mathsf{s}}$ with respect to the relation $\dashv^{\mathsf{s}}$ (Proposition E.3) then we get $\rho \dashv^{\mathsf{s}} \sigma$. Now, since we have shown that all the infinite traces in $\mathsf{sk\Gamma r}(\rho \parallel \sigma)$ are non-definitely-$\mathtt{skp}$, we derive that $f : \rho \dashv^{\mathtt{skp}} \sigma$ by Definition E.1 and Fact E.2. □

We now proceed to the proof of Theorem 6.12, i.e. the proof that

The algorithm **S** is correct and complete

Here we consider orchestrators as explicit terms. Given an orchestrator f we denote by $\mathsf{rt}(f)$ its corresponding (possibly infinite) regular tree.

We start by showing that **SB** is terminating:

Lemma E.15. *For any Γ, ρ and σ, the execution of* $\mathbf{SB}(\Gamma, \rho, \sigma)$ *terminates.*

Proof. All session contracts in the recursive calls of **SB** are sub-expressions of either ρ or σ or of a session contract in a judgement in Γ (which is finite). Since session contracts are regular trees, their sub-expressions are a finite set, so that the test $x : \rho \dashv^{\mathsf{d}} \sigma \in \Gamma$ in clause 2 of **SB** is always successfully reached in case the algorithm does not terminate by clause 1 or the fail clause 6. □

Since the procedure **SB** is the formalisation of a proof search in $\rhd^{\mathsf{inf}}_{\mathsf{sO}}$, we have:

Fact E.16. *If* $f = \mathbf{SB}(\emptyset, \rho, \sigma) \neq \mathbf{fail}$ *then* $\rhd^{\mathsf{inf}}_{\mathsf{sO}} f : \rho \dashv\!\vdash^{\mathsf{d}}_{\_} \sigma$.

The opposite direction of the above implication cannot hold as it is. In fact, consider

$$\rho = \overline{c}.\mathsf{rec}\, x.\overline{a}.\overline{b}.x \quad \sigma = c.\mathsf{rec}\, x.a.b.x \quad f = \langle c, \overline{c}\rangle.\mathsf{rec}\, x.\langle a, \overline{a}\rangle.\langle b, \overline{b}\rangle.\langle a, \overline{a}\rangle.\langle b, \overline{b}\rangle.x.$$

It can be easily checked that $\rhd^{\mathsf{inf}}_{\mathsf{sO}} f : \rho \dashv\!\vdash^{\mathsf{d}}_{\_} \sigma$. We have however that

$$f \neq \mathbf{SB}(\emptyset, \rho, \sigma) = g$$

where $g = \langle c, \overline{c}\rangle.\mathsf{rec}\, x.\langle a, \overline{a}\rangle.\langle b, \overline{b}\rangle.x$.

Of course f and g do represent the very same orchestrating procedure for ρ and σ, since they correspond to the same regular tree, that is $\mathsf{rt}(f) = \mathsf{rt}(g)$.

Lemma E.17. *If* $\rhd^{\inf}_{\mathtt{sO}}\ f : \rho \dashv\!\vdash^{\mathsf{d}}_{\_}\ \sigma$ *then there exists* g *such that* $g = \mathbf{SB}(\emptyset, \rho, \sigma)$ *with* $\mathsf{rt}(f) = \mathsf{rt}(g)$.

Proof. Given a derivation tree for $\rhd^{\inf}_{\mathtt{sO}}\ f : \rho \dashv\!\vdash^{\mathsf{d}}_{\_}\ \sigma$, consider the largest subtree having a conclusion of the form $\Gamma' \rhd^{\inf}_{\mathtt{sO}} \mathsf{rec}x.f' : \rho' \dashv\!\vdash^{\mathsf{d}}_{\_}\ \sigma'$ with $f = F[\mathsf{rec}x.f']$ and such that it has one or more non-nested subtrees of the form $\Gamma'', x : \rho' \dashv\!\vdash^{\mathsf{d}}_{\_}\ \sigma' \rhd^{\inf}_{\mathtt{sO}} f'' : \rho' \dashv\!\vdash^{\mathsf{d}}_{\_}\ \sigma'$, if any, where $f' = F'[f'']$ and $x \in \mathrm{FN}(f'')$ (notice that it could also be the case that $f'' = x$). If no such a subderivation exists, then any rule in the derivation does precisely correspond to a clause of the algorithm **SB** and hence the algorithm returns f. Otherwise, let us take the orchestrator $g' = \mathsf{rec}x.f''$. By the correspondence of the rules of $\rhd^{\inf}_{\mathtt{sO}}$ with the clauses of **SB** we get that $\mathbf{SB}(\Gamma'', \rho, \sigma) = g'$. Moreover, since the rules of System $\rhd^{\inf}_{\mathtt{sO}}$ are such that the proof search is deterministic, we get that either f' can be obtained by a number of unfoldings of the orchestrator g' or $f' = f''\{x/f''\}$ (where the substitution $\{x/f'\}$ has possibly to be performed more than once). This implies in turn that $\mathsf{rt}(f) = \mathsf{rt}(g)$. Moreover, we have that $\mathbf{SB}(\emptyset, \rho, \sigma) = F[g']$. □

In order to get the correctness and completeness of the algorithm **S**, we show that $\dashv\!\vdash^{\mathtt{skp}}_{\mathtt{Orch}}$ can be characterised in terms of $\dashv\!\vdash^{\mathsf{d}}_{\_}$ and the ndsi property.

Lemma E.18. [$f : \rho \dashv\!\vdash^{\mathsf{d}}_{\_}\ \sigma$ *and* f *is ndsi*] *if and only if* $f : \rho \dashv\!\vdash^{\mathtt{skp}}_{\mathtt{Orch}}\ \sigma$.

Proof. The *if* part easily descends from Fact 6.10(1) and Lemma E.13. The *only if* part, instead, follows using Fact E.6 and the definition of $\dashv\!\vdash^{\mathtt{skp}}_{\mathtt{Orch}}$. □

In order Lemma E.17 to be useful, we need to show the following

Lemma E.19. *Let* $f, g \in \mathsf{SkpOrch}$.

$$\mathsf{rt}(f) = \mathsf{rt}(f)\ \textit{implies}\ [f\ \textit{is ndsi} \ \Leftrightarrow\ g\ \textit{is ndsi}\ ]$$

Proof. Immediate by definition of ndsi orchestrator and ndsi sequence of orchestration actions. □

Theorem 6.12 corresponds to the following corollary.

Corollary E.20.

1. *If* $\rho \dashv\!\vdash^{\mathtt{skp}}_{\mathtt{Orch}}\ \sigma$ *then* $\mathbf{S}(\rho, \sigma)$ *terminates and there exists* g *such that* $g = \mathbf{S}(\rho, \sigma) \neq \mathbf{fail}$ *with* $g : \rho \dashv\!\vdash^{\mathtt{skp}}_{\mathtt{Orch}}\ \sigma$.
2. *If* $f = \mathbf{S}(\rho, \sigma) \neq \mathbf{fail}$ *then* $\rho \dashv\!\vdash^{\mathtt{skp}}_{\mathtt{Orch}}\ \sigma$.

Proof.

1. If $\rho \dashv\!\vdash^{\mathtt{skp}}_{\mathtt{Orch}}\ \sigma$ holds, then there exists an f such that $f : \rho \dashv\!\vdash^{\mathtt{skp}}_{\mathtt{Orch}}\ \sigma$. Hence the thesis follows by definition of **S**, Lemmas E.18, E.19, E.17, completeness of System $\rhd^{\inf}_{\mathtt{sO}}$ and the decidability of the ndsi property for orchestrators (Proposition B.12).
2. By definition of **S**, Fact E.16, correctness of System $\rhd^{\inf}_{\mathtt{sO}}$ and Lemma E.18. □

References

[1] K. Honda, V.T. Vasconcelos, M. Kubo, Language primitives and type disciplines for structured communication-based programming, in: ESOP, in: Lect. Notes Comput. Sci., vol. 1381, Springer, 1998, pp. 22–138.
[2] S. Carpineti, G. Castagna, C. Laneve, L. Padovani, A formal account of contracts for Web Services, in: WS-FM, in: Lect. Notes Comput. Sci., vol. 4184, Springer, 2006, pp. 148–162.
[3] C. Laneve, L. Padovani, The must preorder revisited: an algebraic theory for web services contracts, in: CONCUR'07, in: Lect. Notes Comput. Sci., vol. 4703, Springer, 2007, pp. 212–225.
[4] G. Castagna, N. Gesbert, L. Padovani, A theory of contracts for web services, ACM Trans. Program. Lang. Syst. 31 (5) (2009) 19:1–19:61, http://dx.doi.org/10.1145/1538917.1538920.
[5] R. Milner, J. Parrow, D. Walker, A calculus of mobile processes, I & II, Inf. Comput. 100 (1) (1992) 1–77.
[6] F. Barbanera, U. de'Liguoro, Two notions of sub-behaviour for session-based client/server systems, in: PPDP, ACM Press, 2010, pp. 155–164.
[7] F. Barbanera, U. de' Liguoro, Sub-behaviour relations for session-based client/server systems, Math. Struct. Comput. Sci. 25 (2015) 1339–1381, http://dx.doi.org/10.1017/S096012951400005X.
[8] G. Bernardi, M. Hennessy, Modelling session types using contracts, in: SAC'12, ACM, New York, NY, USA, 2012, pp. 1941–1946.
[9] G. Bernardi, M. Hennessy, Modelling session types using contracts, Math. Struct. Comput. Sci. 26 (2016) 510–560, http://dx.doi.org/10.1017/S0960129514000243.
[10] C. Peltz, Web services orchestration and choreography, Computer 36 (10) (2003) 46–52, http://dx.doi.org/10.1109/MC.2003.1236471.
[11] L. Padovani, Contract-based discovery of web services modulo simple orchestrators, Theor. Comput. Sci. 411 (2010) 3328–3347, http://dx.doi.org/10.1016/j.tcs.2010.05.002.
[12] F. Barbanera, S. van Bakel, U. de'Liguoro, Orchestrated session compliance, in: ICE'15, in: EPTCS, vol. 189, Open Publishing Association, 2015, pp. 21–36.

[13] F. Barbanera, U. de' Liguoro, Loosening the notions of compliance and sub-behaviour in client/server systems, in: ICE'14, in: EPTCS, vol. 166, 2014, pp. 94–110.
[14] S. Gay, M. Hole, Subtyping for session types in the pi-calculus, Acta Inform. 42 (2/3) (2005) 191–225, http://dx.doi.org/10.1007/s00236-005-0177-z.
[15] M. Brandt, F. Henglein, Coinductive axiomatization of recursive type equality and subtyping, Fundam. Inform. 33 (4) (1998) 309–338, http://dx.doi.org/10.1007/3-540-62688-3_29.
[16] B. Pierce, D. Sangiorgi, Typing and subtyping for mobile processes, Math. Struct. Comput. Sci. 6 (5) (1996) 409–453.
[17] V. Gapeyev, M.Y. Levin, B.C. Pierce, Recursive subtyping revealed, J. Funct. Program. 12 (6) (2002) 511–548.
[18] O.H. Ibarra, J. Su, Z. Dang, T. Bultan, R. Kemmerer, Counter machines: decidable properties and applications to verification problems, in: MFCS 2000, in: Lect. Notes Comput. Sci., vol. 1893, 2000.
[19] D. Basile, P. Degano, G.L. Ferrari, Automata for analysing service contracts, in: TGC 2014, in: Lect. Notes Comput. Sci., vol. 8902, 2014, pp. 34–50.
[20] D. Basile, P. Degano, G.-L. Ferrari, E. Tuosto, From orchestration to choreography through contract automata, in: ICE'14, in: EPTCS, vol. 166, 2014, pp. 67–85.
[21] D. Brand, P. Zafiropulo, On communicating finite-state machines, J. ACM 30 (2) (1983) 323–342, http://dx.doi.org/10.1145/322374.322380.
[22] P. Deniélou, N. Yoshida, Multiparty session types meet communicating automata, in: ESOP, 2012, pp. 194–213.
[23] L. Clemente, F. Herbreteau, G. Sutre, Decidable topologies for communicating automata with FIFO and bag channels, in: CONCUR'14, in: Lect. Notes Comput. Sci., vol. 8704, 2014.
[24] M. Bartoletti, T. Cimoli, G. Pinna, R. Zunino, Contracts as games on event structures, J. Log. Algebraic Methods Program. 85 (3) (2016) 399–424, http://dx.doi.org/10.1016/j.jlamp.2015.05.001.
[25] F. Barbanera, U. de' Liguoro, A game interpretation of retractable contracts, in: COORDINATION 2016, in: Lect. Notes Comput. Sci., vol. 9686, Springer, 2016, pp. 18–34.
[26] A. Di Stefano, C. Santoro, A peer-to-peer decentralized strategy for resource management in computational grids, Concurr. Comput., Pract. Exp. 19 (9) (2007) 1271–1286, http://dx.doi.org/10.1002/cpe.1110.
[27] F. Messina, G. Pappalardo, D. Rosaci, C. Santoro, G. Sarné, A trust-aware, self-organizing system for large-scale federations of utility computing infrastructures, Future Gener. Comput. Syst. 56 (C) (2016) 77–94, http://dx.doi.org/10.1016/j.future.2015.07.013.