Logical Methods in Computer Science
Vol. 13(3:15)2017, pp. 1–51
https://lmcs.episciences.org/

Submitted Jan. 24, 2017
Published Aug. 25, 2017

RETRACTABILITY, GAMES AND ORCHESTRATORS FOR SESSION CONTRACTS

FRANCO BARBANERA AND UGO DE' LIGUORO

Dipartimento di Matematica e Informatica, University of Catania
e-mail address: barba@dmi.unict.it

Dipartimento di Informatica, University of Torino
e-mail address: ugo.deliguoro@unito.it

Abstract. Session contracts is a formalism enabling to investigate client/server interaction protocols and to interpret session types. We extend session contracts in order to represent outputs whose actual sending in an interaction depends on a third party or on a mutual agreement between the partners. Such contracts are hence adaptable, or as we say "affectible". In client/server systems, in general, compliance stands for the satisfaction of all client's requests by the server. We define an abstract notion of "affectible compliance" and show it to have a precise three-party game-theoretic interpretation. This in turn is shown to be equivalent to a compliance based on interactions that can undergo a sequence of failures and rollbacks, as well as to a compliance based on interactions which can be mediated by an orchestrator. Besides, there is a one-to-one effective correspondence between winning strategies and orchestrators. The relation of subcontract for affectible contracts is also investigated.

The notion of *contract* [11, 14, 12] has been proposed as an abstraction to formally specify and check the behaviour of software systems, and especially of web services. In particular, in the setting of service-oriented architectures, the concept of agreement, often called *compliance*, is of paramount importance while searching for components and ensuring that they will properly interact with each other. The main challenge is that compliance has to meet the contrasting requirements of guaranteeing correctness of interactions w.r.t. certain safety and liveness conditions, while remaining coarse enough to maximize the possibilities of finding compliant components in a library or services through the web.

The main conceptual tool to face the issue is that of relaxing the constraint of a perfect correspondence among contracts through *contract refinement*, also called sub-contract [12, 10] and sub-behaviour [2] relations, that is pre-order relations such that processes conforming to more demanding contracts (which are lower in the pre-order) can be safely substituted

Key words and phrases: contract, session contract, retractability, orchestrator, concurrent game, strategy.

The present paper is a reorganised, revised and extended version of the COORDINATION 2016 conference paper *"A game interpretation of retractable contracts"* [3] by the same authors.

This work was partially supported by the European Union ICT COST Action IC1405 Reversible computation - extending horizons of computing, the Project FIR 1B8C1 of the University of Catania and Project FORMS 2015 of the University of Turin.

DOI:10.23638/LMCS-13(3:15)2017

in contexts allowing more permissive ones. Indeed contract refinement closely resembles subtyping, as it is apparent in the case of session types [13, 2], and it is related to (but doesn't coincide with) observational pre-orders and *must-testing* in process algebra [14, 8].

However, since the first contributions to the theory of contracts [12], a rather different approach has been followed, based on the idea of filtering out certain actions that, althought unmatched on both sides of a binary interaction, can be neglected or prevented by the action of a mediating process called the *orchestrator* [16, 15, 5, 6], without compromising the reaching of the goals of the participants, like the satisfaction of all client requests in a client-server architecture.

Another route for the same purpose is to change the semantics of contracts so that interacting processes can adapt to each other by means of a rollback mechanism: these are the *retractable contracts* proposed in [4]. Although compliance can be decided in advance, interaction among processes exposing retractable contracts undergoes a sequence of failures and backtracks that might be avoided by extracting information from the compliance check.

The contribution of the present paper is to show that the use of orchestrators and retractability are indeed two equivalent approaches to get compliance by adapting, affecting, the behaviour of the partners of a client/server interaction, at least in the case of binary *session contracts* [2, 9]. These are contracts that limit the non-determinism by constraining both external and internal choices to a more regular form, so that they can be looked at as an interpretation of session types into the formalism of contracts [2, 9]. In particular, session contracts can be seen as binary session types without value or channel passing.

The contracts we consider in this paper are session contracts with external choices of outputs that we abstractly look at, in a sense, as the *affectible*, adaptable parts of a contract. These contracts are syntactically the same as the retractable session contracts [4], but instead of adding rollback to the usual contract semantics, we formalise inside an abstract notion of compliance the fact that the actual sending of *affectible*, adaptable outputs can be influenced by an agreement between the interaction partners or by some entity external to the system, in order to make the partners compliant. In particular, *affectible compliance*, i.e. compliance got by means of a (run-time) adaptation of the contracts' behaviours, will be first abstractedly presented as a coinductively defined relation. This relation will be proved later on to be decidable and to coincide both with the retractable compliance relation of [4] involving failures and rollbacks and with the orchestrated compliance, where the (affectible) synchronizations are influenced by elements of a particular class of orchestrators in the sense of [16] and [6].

The essence of this equivalence is that the above mentioned orchestrators correspond to winning strategies in certain concurrent games that naturally model affectible contracts. In [7] the theory of contracts has been grounded on games over event structures among multiple players; applying this framework to affectible contracts, the interaction among a client and a server can be seen as a play in a three-party game. Player A moves according to the normal actions of the client; player B moves according to the normal actions of the server, whereas moves by player C correspond to affectible actions on both sides. The server σ is hence affectible compliant with the client ρ whenever C has a winning strategy in the game with players A and B, where player C wins when she/he succeeds to lead the system $\rho \| \sigma$ to a successful state (the client terminates) or the interaction proceeds indefinitely without deadlocking.

The payoff of the game theoretic interpretation is that there is a precise correspondence between winning strategies for player C and elements of a class of orchestrators in the sense

of [16] and [6]. Such a correspondence is of interest on its own, since strategies are abstract entities while orchestrators are terms of a process algebra and concrete witnesses of the agreement among participants of a session. Moreover, we can decide whether a client/server pair is affectible compliant by means of an algorithm that synthesizes an orchestrator, if any, or reports failure.
We also show that there is a one-to-one correspondence between orchestrators and derivations in the formal system axiomatizing the relation of affectible compliance.

The substitutability relation (affectible subcontract) on servers, induced by the relation of affectible compliance, can be defined as for the usual subcontract relations. Its decidability, however has to be proved in a direct way: the introduction of adaptability implies that decidability of the subcontract relation cannot be simply inferred from decidability of compliance. It descends instead from correctness and termination of the proof recontruction algorithm for the formal system for the affectible subcontract relation. Moreover, we shall show how a derivation in such a formal system does correspond in an effective way to a functor on orchestrators. In particular, if σ is proved to be a subcontract of σ', the functor transforms any orchestrator making σ compliant with a client ρ into an orchestrator making σ' compliant with ρ.

The present paper is a reorganised, revised and extended version of [3], where most of the proofs had been omitted and where the correspondence between derivations for the compliance relation and orchestrators was not present. Besides, the relation of subcontract was not investigated in [3], as well as the correspondence between derivations for the subcontract relation and orchestrator functors.

Overview of the paper. In Section 1 we define affectible session contracts and the abstract notion of compliance on them. In Section 2 we recall the notion of multi-player game from [7] based on event structures. We then show how it is possible to interpret a client/server system $\rho\|\sigma$ with a three-players game $\mathcal{G}_{\rho\|\sigma}$ by means of a turn-based operational semantics. Sections 3 will be devoted to the formalization of the notion of interactions with rollbacks and the related notion of retractable compliance. Orchestrators and orchestrated interactions for affectible contracts will be defined in Section 4, together with the notion of orchestrated compliance. In 5 an axiomatization for affectible compliance will be provided and we shall show how all the above mentioned notions are related with each other: the Main Theorem I will essentially state that the abstract notion of affectible compliance and its game-theoretic interpretation are but an abstract representation of both retractable and orchestrated compliance. The Main Theorem II will show instead how it is effectively possible to get derivations, winning strategies and orchestrators out of each other. The definition of the subcontract relation, its axiomatization and decidability, together with the correspondence between subcontract derivations and orchestrators functors will be the topic of Section 6. A Conclusion and Future Work section (Section 7) will be the last one before some appendices. We shall use a simple working example through the various sections in order to clarify the notions we introduce. Many proofs and accessory formalisms will be detailed in the appendices at the end of the paper.

1. Affectible session contracts

Affectible session contracts (affectible contracts for short) stem from *session contracts* [2, 8][1]. With respect to session contracts, affectible contracts add the affectible output construct, which is called retractable output in [4]. The affectible output operator aims at representing points where the client/server interaction can be influenced by a third party or by an *agreement* between the two partners; consequently it is natural to use the CCS external choice operator as it is the case of the input branching (which is always affectible). Outputs in an internal choice are regarded as unaffectible actions and treated as unretractable in the setting of [4].

Definition 1.1 (Affectible session contracts)**.** Let $\mathcal{N}$ *(names)* be some countable set of symbols and let $\overline{\mathcal{N}} = \{\, \overline{a} \mid a \in \mathcal{N} \,\}$ *(co-names)*, with $\mathcal{N} \cap \overline{\mathcal{N}} = \emptyset$.
The set ASC of **affectible session contracts** is defined as the set of the **closed** (with respect to the binder rec) expressions generated by the following grammar,

$$
\begin{array}{rcll}
\sigma, \rho & := & \mid \mathbf{1} & \text{(success)} \\
& & \mid \sum_{i \in I} a_i.\sigma_i & \text{(input)} \\
& & \mid \sum_{i \in I} \overline{a}_i.\sigma_i & \text{(affectible output)} \\
& & \mid \bigoplus_{i \in I} \overline{a}_i.\sigma_i & \text{(unaffectible output)} \\
& & \mid x & \text{(variable)} \\
& & \mid \mathsf{rec}\, x.\sigma & \text{(recursion)}
\end{array}
$$

where

- I is non-empty and finite;
- names and co-names in choices are pairwise distinct;
- σ is not a variable in $\mathsf{rec}\, x.\sigma$.

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

Notation 1. As usual, when $I = \{\, 1, .., n \,\}$, we shall indifferently use the notations $a_1.\sigma_1 + \cdots + a_n.\sigma_n$ and $\Sigma_{i \in I} a_i.\sigma_i$, as well as $\overline{a}_1.\sigma_1 \oplus \cdots \oplus \overline{a}_n.\sigma_n$ and $\bigoplus_{i \in I} \overline{a}_i.\sigma_i$.
We also shall write $\alpha_k.\sigma_k + \sigma'$ to denote $\Sigma_{i \in I} \alpha_i.\sigma_i$ where $k \in I$ and $\sigma' = \Sigma_{i \in (I \setminus \{\, k \,\})} \alpha_i.\sigma_i$. Similarly for the internal choice.
We assume I never to be a singleton in $\Sigma_{i \in I} \overline{a}_i.\sigma_i$. This means that a term like $a_k.\sigma$ has to be unambiguously read as $\bigoplus_{i \in \{\, k \,\}} \overline{a}_i.\sigma$ and not as $\Sigma_{i \in \{\, k \,\}} \overline{a}_i.\sigma$

Recursion is guarded and hence contractive in the usual sense. Unless stated otherwise, 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 shall write $a + b$ for $a.\mathbf{1} + b.\mathbf{1}$. Affectible contracts will be considered modulo commutativity of internal and external choices.

The notion of compliance for (client-server) pairs of contracts is usually defined by means of an LTS. A *client-server system* (a *system* for short) is a pair $\rho \parallel \sigma$ of contracts, where ρ plays the role of client and σ of server; let the relation $\rho \parallel \sigma \rightarrow \rho' \parallel \sigma'$ represent a communication step resulting into the new system ρ', σ'; nowρ and σ are compliant if $\rho \parallel \sigma \stackrel{*}{\longrightarrow} \rho' \parallel \sigma' \not\rightarrow$ implies $\rho = \mathbf{1}$. Then one studies the properties of the compliance relation, possibly with reference to the contract syntax alone (see e.g. [2]). With affectible contracts,

[1]The name used in [2] was actually *session behaviours*.

however, the semantics of $\|$ is more complex as it involves a form of backtracking. For the present exposition we prefer to move from an abstract coinductive definition for the affectible compliance relation. Later on we shall prove that defining compliance out of the retractable operational semantics (Section 3) is equivalent to defining it out of the orchestrated operational semantics (Section 4) and that both approaches are equivalent to the abstract affectible compliance below.

Definition 1.2. The $\mathbb{A}$ffectible $\mathbb{C}$ompliance relation $\mathbb{AC} \subseteq \mathsf{ASC} \times \mathsf{ASC}$ is coinductively defined as follows. Let $\rho, \sigma \in \mathsf{ASC}$. Then $\rho \,\mathbb{AC}\, \sigma$ if one of the following conditions holds:

(1) $\rho = \mathbf{1}$;
(2) $\rho = \sum_{i\in I} \alpha_i.\rho_i$, $\sigma = \sum_{j\in J} \overline{\alpha}_j.\sigma_j$ and $\exists k \in I \cap J.\ \rho_k \,\mathbb{AC}\, \sigma_k$;
(3) $\rho = \bigoplus_{i\in I} \overline{a}_i.\rho_i$, $\sigma = \sum_{j\in J} a_j.\sigma_j$, $I \subseteq J$ and $\forall k \in I.\ \rho_k \,\mathbb{AC}\, \sigma_k$;
(4) $\rho = \sum_{i\in I} a_i.\rho_i$, $\sigma = \bigoplus_{j\in J} \overline{a}_j.\sigma_j$, $I \supseteq J$ and $\forall k \in J.\ \rho_k \,\mathbb{AC}\, \sigma_k$.

Let us informally describe the sense of the previous definition. Consider a system $\rho \,\|\, \sigma$ of affectible contracts. An unaffectible communication between ρ and σ is a synchonization involving unaffectible outputs and inputs. In such a synchronization, any of the unaffectible outputs exhibited by one of the two contracts can be expected, since it depends on an internal decision of the process whose behaviour is abstractedly represented by the contract.

An affectible communication, instead, is a synchronization involving affectible outputs and inputs. Affectable outputs are intended not to depend on internal decisions, but to be *influenced from the outside* in order to enable the system not to get to any stuck state. In client/server interactions there is a bias towards the client. So a stuck state can be interpreted by any pair $\rho \,\|\, \sigma$ where $\rho \neq \mathbf{1}$ but no communication is possible.
So, in a system $\rho \,\|\, \sigma$, the server σ is *affectible-compliant* with the client ρ if either $\rho = \mathbf{1}$, namely the client has successfully terminated; or all unaffectible communications of the system $\rho \,\|\, \sigma$ lead to compliant systems; or there exists an affectible communication leading to a compliant system.

In the above informal description, the worlds " influenced from the outside" are rather abstract; they can be made concrete either via the characterization in terms of retractable computations, as done in Section 3, or in terms of orchestrated interactions as done in Section 4.

Example 1.3. *Let us consider the following example from* [4]. *A* Buyer *is looking for a bag (*$\overline{\texttt{bag}}$*) or a belt (*$\overline{\texttt{belt}}$*); she will decide how to pay, either by credit card (*$\overline{\texttt{card}}$*) or by cash (*$\overline{\texttt{cash}}$*), after knowing the* price *from the* Seller*:*

$$\mathsf{Buyer} = \overline{\texttt{bag}}.\texttt{price}.(\overline{\texttt{card}} \oplus \overline{\texttt{cash}}) + \overline{\texttt{belt}}.\texttt{price}.(\overline{\texttt{card}} \oplus \overline{\texttt{cash}})$$

The Seller *does not accept credit card payments for items of low price, like belts, but only for more expensive ones, like bags:*

$$\mathsf{Seller} = \texttt{belt}.\overline{\texttt{price}}.\texttt{cash} + \texttt{bag}.\overline{\texttt{price}}.(\texttt{card} + \texttt{cash})$$

From the previous definition it is not difficult to check that Buyer $\mathbb{AC}$ Seller*.*

Remark 1.4. Notice that, unlike for session contracts, we have no notion of syntactical duality for affectible contracts, since it would not be definable in a natural way (i.e. in such a way duality be involutive and a contract be always compliant with its dual). In fact for affectible contracts we should define $\overline{a + \overline{b}}$ either as $a + b$ or as $\overline{a} \oplus \overline{b}$. In both cases, however, we would lose the involutive property of the duality operator. In the first case we would

get $\overline{\overline{\overline{a}+\overline{b}}} = \overline{a+b} = \overline{a} \oplus \overline{b}$ (since the duality operator should reasonably be a conservative extension over session contracts, and hence $\overline{a+b} = \overline{a} \oplus \overline{b}$). A similar problem would arise by defining $\overline{\overline{a}+\overline{b}} = \overline{a} \oplus \overline{b}$.

2. Game-theoretic interpretation of affectible contracts

Following [7] we interpret client-server systems of affectible contracts as games over event structures. This yields a game-theoretic interpretation of affectible compliance that will be of use to relate this last notion to both retractable and orchestrated compliance.

For the reader's convenience we briefly recall the basic notions of event structure and game associated to an LTS.

Definition 2.1 (Event structure [17])**.** Let $\mathbf{E}$ be a denumerable universe of *events*, ranged over by $e, e', \ldots$, and let $\mathbf{A}$ be a universe of *action labels*, ranged over by $\alpha, \alpha', \beta, \ldots$. Besides, let $E \subseteq \mathbf{E}$ and let $\# \subseteq E \times E$ be an irreflexive and symmetric relation (called *conflict relation*).

(1) The predicate CF on sets $X \subseteq E$ and the set Con of finite *conflict-free* sets are defined by
$$CF(X) = \forall e, e' \in X. \neg(e \# e') \qquad Con = \{\, X \subseteq_{fin} E \mid CF(X) \,\}$$
(2) An *event structure* is a quadruple $\mathcal{E} = (E, \#, \vdash, l)$ where
- $\vdash \subseteq Con \times E$ is a relation such that $sat(\vdash) = \vdash$ (i.e. $\vdash$ is *saturated*), where $sat(\vdash) = \{\, (Y, e) \mid X \vdash e \ \ \& \ \ X \subseteq Y \in Con \,\}$;
- $l : E \to \mathbf{A}$ is a labelling function.

Given a set E of events, E^∞ denotes the set of sequences (both finite and infinite) of its elements. We denote by $\boldsymbol{e} = \langle e_0 e_1 \cdots \rangle$ (or simply $e_0 e_1 \cdots$) a sequence of events[2]. Given $\boldsymbol{e}$, we denote by $\widehat{\boldsymbol{e}}$ the set of its elements, by $|\boldsymbol{e}|$ its length (either a natural number or ∞) and by $\boldsymbol{e}_{/i}$, for $i < |\boldsymbol{e}|$, the subsequence $\langle e_0 e_1 \cdots e_{i-1} \rangle$ of its first i elements. Given a set X we denote by $|X|$ its cardinality. $\mathbb{N}$ is the set of natural numbers. The symbol '$\rightharpoonup$' will be used, as usual, to denote partial mappings.

Definition 2.2 (LTS over configurations [7])**.** Given an event structure $\mathcal{E} = (E, \#, \vdash, l)$, we define the LTS $(\mathcal{P}_{\text{fin}}(E), E, \to_{\mathcal{E}})$ (where $\mathcal{P}_{\text{fin}}(E)$ is the set of states, E the set of labels and the labelled transition $\to_{\mathcal{E}}$) as follows:
$$C \xrightarrow{e}_{\mathcal{E}} C \cup \{\, e \,\} \quad \text{if} \quad C \vdash e,\ e \notin C \text{ and } CF(C \cup \{\, e \,\})$$
We shall omit the subscript in $\to_{\mathcal{E}}$ when clear from the context. For sake of brevity we shall often denote an LTS $(S, L, \to\,)$ by simply $(S, \to\,)$ or $\to$.

Given an LTS $(S, \to)$ and a state $s \in S$, we denote by $\mathsf{Tr}(s, \to)$ the set of the (finite or infinite) traces in $(S, \to)$ starting in s, that is $\mathsf{Tr}(s, \to) = \{\, s_0 s_1 \cdots \mid s_0 = s, s_i \to\ s_{i+1} \,\}$.

[2]Differently than in [7], we use the notation $\boldsymbol{e}$ for sequences instead of σ, which refers to a contract here.

2.1. **Multi-player games.** All the subsequent definitions and terminology are from [7], except in the case of games that we call "multi-player" instead of "contracts", which would be confusing in the present setting.

A set of participants (players) to a game will be denoted by $\mathfrak{P}$, whereas the universe of participants is denoted by $\mathfrak{P}_{\mathfrak{U}}$. We shall use A, B,... as variables ranging over $\mathfrak{P}$ or $\mathfrak{P}_{\mathfrak{U}}$. The symbols A, B, ... will denote particular elements of $\mathfrak{P}$ or $\mathfrak{P}_{\mathfrak{U}}$. We assume that each event is associated to a player by means of a function $\pi : \mathbf{E} \to \mathfrak{P}_{\mathfrak{U}}$. Moreover, given $A \in \mathfrak{P}_{\mathfrak{U}}$ we define $\mathbf{E}_A = \{\, e \in \mathbf{E} \mid \pi(e) = A \,\}$.

Definition 2.3 (Multi-player game)**.**

(1) A *game* $\mathcal{G}$ is a pair $(\mathcal{E}, \Phi)$ where $\mathcal{E} = (E, \#, \vdash, l)$ is an event structure and $\Phi : \mathfrak{P}_{\mathfrak{U}} \rightharpoonup E^\infty \to \{\, -1, 0, 1 \,\}$ associates each participant and trace with a *payoff*. Moreover, for all $X \vdash e$ in $\mathcal{E}$, $\Phi(\pi(e))$ is defined. We say that $\mathcal{G}$ is a game with participants $\mathfrak{P}$ whenever ΦA is defined for each player A in $\mathfrak{P}$.

(2) A *play* of a game $\mathcal{G} = (\mathcal{E}, \Phi)$ is a (finite or infinite) trace of $(\emptyset, \to_{\mathcal{E}})$ i.e. an element of $\mathsf{Tr}(\emptyset, \to_{\mathcal{E}})$.

Definition 2.4 (Strategy and conformance)**.**

(1) A *strategy* Σ for a participant A in a game $\mathcal{G}$ is a function which maps each finite play $\boldsymbol{e}{=}\langle e_0 \cdots e_n \rangle$ to a (possibly empty) subset of $\mathbf{E}_A$ such that: $e{\in}\Sigma(\boldsymbol{e}) \;\Rightarrow\; \boldsymbol{e}e$ is a play of $\mathcal{G}$.

(2) A play $\boldsymbol{e} = \langle e_0 e_1 \cdots \rangle$ *conforms* to a strategy Σ for a participant A in $\mathcal{G}$ if, for all $0 \leq i < |\boldsymbol{e}|$, $e_i \in \mathbf{E}_A \;\Rightarrow\; e_i \in \Sigma(\boldsymbol{e}_{/i})$.

In general there are neither a turn rule nor alternation of players, similarly to concurrent games in [1]. A strategy Σ provides "suggestions" to some player on how to legally move continuing finite plays (also called "positions" in game-theoretic literature). But Σ may be ambiguous at some places, since $\Sigma(\boldsymbol{e})$ may contain more than an event; in fact it can be viewed as a partial mapping which is undefined when $\Sigma(\boldsymbol{e}) = \emptyset$.

We refer to [7] for the general definition of winning strategy for multi-player games (briefly recalled also in Remark 2.15 below), since it involves the conditions of fairness and innocence, which will be trivially satisfied in our interpretation of affectible client-server systems, where the notion of winning strategy corresponds to the one that will be given in Definition 2.12.

We define now the notion of *univocal strategy*. When showing the equivalence between the various notions of compliance and the existence of winning strategies, we shall restrict to univocal strategies for the sake of simplicity.

Definition 2.5 (Univocal strategies)**.** Σ is *univocal* if $\forall \boldsymbol{e}.\ |\Sigma(\boldsymbol{e})| \leq 1$.

2.2. **Turn-based operational semantics and compliance.** Toward the game theoretic interpretation of a client-server system $\rho \,\|\, \sigma$, we introduce an operational semantics of affectible contracts, making explicit the idea of a three-player game. We interpret the internal choices and the input actions of the client as moves of a player A and the internal choices and the input actions of the server as moves of a player B. The synchronisations due to affectible choices are instead interpreted as moves of the third player C.

From a technical point of view this is a slight generalization and adaptation to our scenario of the turn-based semantics of "session types" in [7], §5.2. The changes are needed both because we have three players instead of two, and because session types are just session contracts, that is affectible contracts without affectible outputs.

$$\oplus_{i\in I}\overline{a}_i.\rho_i \parallel \tilde{\sigma} \xrightarrow{\mathsf{A}:\overline{a}_k} [\overline{a}_k]\rho_k \parallel \tilde{\sigma} \qquad \Sigma_{i\in I}a_i.\rho_i \parallel [\overline{a}_k]\sigma \xrightarrow{\mathsf{A}:a_k} \rho_k \parallel \sigma$$

$$\tilde{\rho} \parallel \oplus_{i\in I}\overline{a}_i.\sigma_i \xrightarrow{\mathsf{B}:\overline{a}_k} \tilde{\rho} \parallel [\overline{a}_k]\sigma_k \qquad [\overline{a}_k]\rho \parallel \Sigma_{i\in I}a_i.\sigma_i \xrightarrow{\mathsf{B}:a_k} \rho \parallel \sigma_k$$

$$\overline{a}.\rho + \rho' \parallel a.\sigma + \sigma' \xrightarrow{\mathsf{C}:a} \rho \parallel \sigma \qquad a.\rho + \rho' \parallel \overline{a}.\sigma + \sigma' \xrightarrow{\mathsf{C}:a} \rho \parallel \sigma$$

$$\overline{a}.\rho + \rho' \parallel a.\sigma \xrightarrow{\mathsf{C}:a} \rho \parallel \sigma \qquad a.\rho \parallel \overline{a}.\sigma + \sigma' \xrightarrow{\mathsf{C}:a} \rho \parallel \sigma$$

$$\mathbf{1} \parallel \tilde{\rho} \xrightarrow{\mathsf{C}:\checkmark} \mathbf{0} \parallel \tilde{\rho}$$

where $(k \in I)$

Figure 1: Turn-based operational semantics of turn-based configurations

Definition 2.6 (Single-buffered ASC)**.** The set $\mathsf{ASC}^{[]}$ of *single-buffered* affectible contracts is defined by

$$\mathsf{ASC}^{[]} = \mathsf{ASC} \cup \{\, \mathbf{0} \,\} \cup \{\, [\overline{a}]\sigma \mid \overline{a} \in \overline{\mathcal{N}}, \sigma \in \mathsf{ASC} \,\}.$$

We use the symbols $\tilde{\rho}, \tilde{\sigma}, \tilde{\rho}', \tilde{\sigma}' \ldots$ to denote elements of $\mathsf{ASC}^{[]}$. A *turn-based configuration* (a configuration for short) is a pair $\tilde{\rho} \parallel \tilde{\sigma}$, where $\tilde{\rho}, \tilde{\sigma} \in \mathsf{ASC}^{[]}$.

As in [7], we have added the "single buffered" contracts $[\overline{a}]\sigma$ to represent the situation in which $\overline{a}$ is the only output offered after an internal choice. Since the actual synchronization takes place in a subsequent step, $\overline{a}$ is "buffered" in front of the continuation σ. Such a technical device is adopted for two separate motivations. First, since we wish our game interpretation of configurations to extend the one provided in [7]. Second, since by means of that we shall manage to get easier proofs. In fact, using the following turn-based operational semantics any move of player A, unless getting to a stuck state, is necessarily followed by a move of B; and any move of player B is necessarily preceded by a move of A.

Definition 2.7 (Turn-based operational semantics of configurations)**.** Let $\mathbf{tbAct} = \{\mathsf{A}, \mathsf{B}, \mathsf{C}\} \times (\mathbf{Act} \cup \{\checkmark\})$, where $\mathsf{A}, \mathsf{B}, \mathsf{C}$ are particular elements of $\mathfrak{P}_{\mathfrak{U}}$.
In Figure 1 we define the LTS $\longrightarrow\!\!\!\!\!\rightarrow$ over turn-based configurations, with labels in $\mathbf{tbAct}$.

An element $\mathsf{A}{:}\overline{a}$ (resp. $\mathsf{A}{:}\overline{a}$) of $\mathbf{tbAct}$ represents an output (resp. input) action performed by player A. Similarly for B. An element $\mathsf{C}{:}a$ represents one of the possible way the player C can affect the interaction between affectible-outputs and inputs. An element $\mathsf{C}{:}\checkmark$ represents instead a "winning" move. We use the symbols $\beta, \beta', \ldots$ to denote elements of $\mathbf{tbAct}$.

Comparing $\longrightarrow\!\!\!\!\!\rightarrow$ with the usual LTS for session contracts [2, 9], we observe that $[\overline{a}]\sigma$ is a duplicate of $\overline{a}.\sigma$, with the only difference that now there is a redundant step in $\oplus_{i\in I}\overline{a}_i.\rho_i \parallel \tilde{\sigma} \xrightarrow{\mathsf{A}:\overline{a}_k} [\overline{a}_k]\rho_k \parallel \tilde{\sigma}$ when I is the singleton $\{\, k \,\}$. Also, besides the reductions concerning the affectible choices, we have the new reduction $\mathbf{1} \parallel \tilde{\rho} \xrightarrow{\mathsf{C}:\checkmark} \mathbf{0} \parallel \tilde{\rho}$ to signal when player C wins.
Let $\boldsymbol{\beta} = \langle \beta_1 \cdots \beta_n \rangle \in \mathbf{tbAct}^*$. We shall use the notation $\xrightarrow{\boldsymbol{\beta}}\!\!\!\!\!\rightarrow = \xrightarrow{\beta_1}\!\!\!\!\!\rightarrow \circ \cdots \circ \xrightarrow{\beta_n}\!\!\!\!\!\rightarrow$.

The relation of turn-based compliance on single-buffered affectible contracts (and hence on affectible contracts) is defined coinductively using the operational semantics formalised by the LTS $\longrightarrow\!\!\!\!\!\rightarrow$, as follows.

Definition 2.8 (Turn-Based Compliance Relation $\dashv_{\mathsf{tb}}$)**.**

(1) Let $\mathcal{H} : \mathcal{P}(\mathsf{ASC}^{[]} \times \mathsf{ASC}^{[]}) \to \mathcal{P}(\mathsf{ASC}^{[]} \times \mathsf{ASC}^{[]})$ be such that, for any $\mathcal{R} \subseteq \mathsf{ASC}^{[]} \times \mathsf{ASC}^{[]}$, we have $(\tilde{\rho}, \tilde{\sigma}) \in \mathcal{H}(\mathcal{R})$ if the following conditions hold:
   (a) $\tilde{\rho} \parallel \tilde{\sigma} \not\twoheadrightarrow$ implies $\rho = \mathbf{0}$;
   (b) $\forall \tilde{\rho}', \tilde{\sigma}'.\ [\ \tilde{\rho} \parallel \tilde{\sigma} \xrightarrow{\beta}\!\!\!\!\twoheadrightarrow\ \tilde{\rho}' \parallel \tilde{\sigma}'$ implies $\tilde{\rho}'\ \mathcal{R}\ \tilde{\sigma}'\ ]$, where $\beta \in \{\mathsf{A}{:}a, \mathsf{A}{:}\overline{a}, \mathsf{B}{:}a, \mathsf{B}{:}\overline{a} \mid a \in \mathcal{N}\}$;
   (c) $\exists a \in \mathcal{N}.\tilde{\rho} \parallel \tilde{\sigma} \xrightarrow{\mathsf{C}:a}\!\!\!\!\twoheadrightarrow$ implies $\exists \tilde{\rho}', \tilde{\sigma}', a.\ [\tilde{\rho} \parallel \tilde{\sigma} \xrightarrow{\mathsf{C}:a}\!\!\!\!\twoheadrightarrow\ \tilde{\rho}' \parallel \tilde{\sigma}'$ and $\tilde{\rho}'\ \mathcal{R}\ \tilde{\sigma}']$;

(2) A relation $\mathcal{R} \subseteq \mathsf{ASC}^{[]} \times \mathsf{ASC}^{[]}$ is a *turn-based compliance relation* if $\mathcal{R} \subseteq \mathcal{H}(\mathcal{R})$. $\dashv_{\mathsf{tb}}$ is the greatest solution of the equation $X = \mathcal{H}(X)$, that is $\dashv_{\mathsf{tb}} = \nu\mathcal{H}$.

(3) For $\rho, \sigma \in \mathsf{ASC}$, we say that σ is *turn-based compliant* with ρ if $\rho \dashv_{\mathsf{tb}} \sigma$.

We can now show that turn-based compliance restricted to contracts in ASC and affectible compliance do coincide.

Proposition 2.9 ($\mathbb{AC}$ and $\dashv_{\mathsf{tb}}$ equivalence.)**.** *Let* $\rho, \sigma \in \mathsf{ASC}$.

$$\rho \dashv_{\mathsf{tb}} \sigma \ \Leftrightarrow\ \rho\, \mathbb{AC}\, \sigma.$$

Proof. See Appendix A □

It will be relatively easy to get a game interpretation of the relation $\dashv_{\mathsf{tb}}$. So the proposition 2.9 will be used to get a game interpretation for the relation $\mathbb{AC}$.

2.3. **Three-player game interpretation for ASC client/server systems.** Using the turn-based semantics, we associate to any client/server system an event structure, and then a three-player game[3], extending the treatment of session types with two-player games in [7]. For our purposes we just consider the LTS of a given client/server system instead of an arbitrary one.

Definition 2.10 (ES of affectible-contracts systems)**.** Let $\rho \parallel \sigma$ be a client/server system of affectible contracts. We define the event structure $[\![\rho \parallel \sigma]\!] = (E, \#, \vdash, l)$, where

- $E = \{\, (n, \beta) \mid n \in \mathbb{N}, \beta \in \mathbf{tbAct} \,\}$
- $\# = \{\, ((n, \beta_1), (n, \beta_2)) \mid n \in \mathbb{N}, \beta_1, \beta_2 \in \mathbf{tbAct}, \beta_1 \neq \beta_2 \,\}$
- $\vdash\ =\ sat(\vdash_{\rho\parallel\sigma})$
  where $\vdash_{\rho\parallel\sigma} = \{\, (X, (n, \beta)) \mid \rho \parallel \sigma \xrightarrow{\mathsf{snd}(X)}\!\!\!\!\twoheadrightarrow\ \tilde{\rho}' \parallel \tilde{\sigma}' \xrightarrow{\beta}\!\!\!\!\twoheadrightarrow\ \text{ and } n = |X| + 1 \,\}$
- $l(n, \beta) = \beta$

where the partial function $\mathsf{snd}(\text{-})$ maps any $X = \{\, (i, \beta_i) \,\}_{i=1..n}$ to $\langle \beta_1 \cdots \beta_n \rangle$, and it is undefined over sets not of the shape of X.

Events in $[\![\rho \parallel \sigma]\!]$ are actions in $\mathbf{tbAct}$ paired with time stamps. Two events are in conflict if different actions should be performed at the same time, so that configurations must be linearly ordered w.r.t. time. The relation $X \vdash_{\rho\parallel\sigma} (n, \beta)$ holds if X is a trace in the LTS of $\rho \parallel \sigma$ of length $n - 1$; therefore the enabling $Y \vdash (n, \beta)$ holds if and only if Y includes a trace of length $n - 1$ that can be prolonged by β, possibly including (n, β) itself and any other action that might occur after β in the LTS.

[3] Such interpretation is called *semantic-based* in [7] and it applies quite naturally to our context. Instead the *syntax-based* approach (which is equivalent to the semantic-based one in a two-players setting; see [7] §5.3.2) cannot be straightforwardly extended to a three-player game.

Example 2.11. By the above definition, $\vdash_{\mathsf{Buyer}\|\mathsf{Seller}}$ in $[\![\mathsf{Buyer} \,\|\, \mathsf{Seller}]\!]$ corresponds to

$$\begin{array}{l}\{\quad \emptyset \vdash_{\mathsf{Buyer}\|\mathsf{Seller}} (1,(\mathsf{C}{:}\mathtt{belt})),\ \emptyset \vdash_{\mathsf{Buyer}\|\mathsf{Seller}} (1,(\mathsf{C}{:}\mathtt{bag})),\\ \quad \{(1,(\mathsf{C}{:}\mathtt{belt}))\} \vdash_{\mathsf{Buyer}\|\mathsf{Seller}} (2,(\mathsf{B}{:}\overline{\mathtt{price}})), \{(1,(\mathsf{C}{:}\mathtt{bag}))\} \vdash_{\mathsf{Buyer}\|\mathsf{Seller}} (2,(\mathsf{B}{:}\overline{\mathtt{price}})),\\ \quad \{(1,(\mathsf{C}{:}\mathtt{belt})),(2,(\mathsf{Seller}{:}\overline{\mathtt{price}}))\} \vdash_{\mathsf{Buyer}\|\mathsf{Seller}} (3,(\mathsf{A}{:}\mathtt{price})),\ldots\\ \quad \ldots X_1 \vdash_{\mathsf{Buyer}\|\mathsf{Seller}} (6,(\mathsf{C},\checkmark)) \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \}\end{array}$$

where $X_1 = \{(1,(\mathsf{C}{:}\mathtt{bag})),(2,(\mathsf{B}{:}\overline{\mathtt{price}})),(3,(\mathsf{A}{:}\mathtt{price})),(4,(\mathsf{A}{:}\overline{\mathtt{cash}})),(5,(\mathsf{B}{:}\mathtt{cash}))\}$. The $\vdash_{\rho\|\sigma}$ of this simple example is finite. It is not so in general for systems with recursive contracts.

The following definition is a specialisation of Definitions 4.6 and 4.7 in [7].
We use $\mathsf{MaxTr}(s,\to)$ and $\mathsf{FinMaxTr}(s,\to)$ to denote the set of maximal traces and finite maximal traces, respectively, of $\mathsf{Tr}(s,\to)$.

Definition 2.12 (Game $\mathcal{G}_{\rho\|\sigma}$)**.** Given $\rho,\sigma \in \mathsf{ASC}$, we define the game $\mathcal{G}_{\rho\|\sigma}$ as $([\![\rho \,\|\, \sigma]\!], \Phi)$, where: $\pi(n,\beta) = A$ if $\beta = A{:}\alpha$, ΦA is defined only if $A \in \{\mathsf{A},\mathsf{B},\mathsf{C}\}$ and

$$\Phi A\boldsymbol{e} = \begin{cases} 1 & \text{if } \mathbf{P}(A,\boldsymbol{e}) \\ -1 & \text{otherwise}\end{cases}$$

where $\mathbf{P}(A,\boldsymbol{e})$ holds whenever

$$\boldsymbol{e} \in \mathsf{Tr}(\emptyset,\to_{[\![\rho\|\sigma]\!]}) \ \&\ [\boldsymbol{e} \in \mathsf{FinMaxTr}(\emptyset,\to_{[\![\rho\|\sigma]\!]}) \ \Rightarrow\ \exists \boldsymbol{e}', n.\, \boldsymbol{e} = \boldsymbol{e}'(n,(A{:}\checkmark))].$$

In words $\mathbf{P}(A,\boldsymbol{e})$ holds if whenever $\boldsymbol{e}$ is a maximal trace in the LTS $\mathsf{Tr}(\emptyset,\to_{[\![\rho\|\sigma]\!]})$ which is finite, then it ends by the event $(n,(A{:}\checkmark)$, where n is just a time stamp, $A{:}\checkmark$ is the successful action performed by the participant A. Note that if $\boldsymbol{e}$ is not a trace of the LTS or it is not maximal, then $\mathbf{P}(A,\boldsymbol{e})$ trivially holds.

A player A *wins* in the sequence of events $\boldsymbol{e}$ if $\Phi A\,\boldsymbol{e} > 0$. A strategy Σ for player A is *winning* if A wins in all plays conforming to Σ.

Note that, if an element $\boldsymbol{e}$ of $\mathsf{Tr}(\emptyset,\to_{[\![\rho\|\sigma]\!]})$ is infinite, $\mathbf{P}(A,\boldsymbol{e})$ holds for any A since the implication is vacuously satisfied. If $\boldsymbol{e}$ is finite, instead, $\mathbf{P}(A,\boldsymbol{e})$ holds only in case the last element of the sequence $\boldsymbol{e}$ is of the form $(n,(A{:}\checkmark))$.

Example 2.13. For the game $\mathcal{G}_{\mathsf{Buyer}\,\|\,\mathsf{Seller}}$, it is possible to check that, for instance,

$$\Phi\mathsf{C}\boldsymbol{s_1} = 1,\ \Phi\mathsf{A}\boldsymbol{s_1} = -1,\ \Phi\mathsf{B}\boldsymbol{s_2} = -1,\ \Phi\mathsf{C}\boldsymbol{s_3} = -1$$

where

$$\begin{array}{l}\boldsymbol{s_1} = (1,(\mathsf{C}{:}\mathtt{bag}))(2,(\mathsf{B}{:}\overline{\mathtt{price}}))(3,(\mathsf{A}{:}\mathtt{price}))(4,(\mathsf{A}{:}\overline{\mathtt{cash}}))(5,(\mathsf{B}{:}\mathtt{cash}))(6,(\mathsf{C},\checkmark)),\\ \boldsymbol{s_2} = (4,(\mathsf{A}{:}\mathtt{bag}))(1,(\mathsf{C}{:}\overline{\mathtt{price}}))\\ \boldsymbol{s_3} = (1,(\mathsf{C}{:}\mathtt{bag}))(2,(\mathsf{B}{:}\overline{\mathtt{price}}))(3,(\mathsf{A}{:}\mathtt{price}))(4,(\mathsf{A}{:}\overline{\mathtt{cash}}))(5,(\mathsf{B}{:}\mathtt{cash}))\end{array}$$

Let us define a particular strategy $\widetilde{\Sigma}$ for C in $\mathcal{G}_{\mathsf{Buyer}\,\|\,\mathsf{Seller}}$ as follows:

$$\widetilde{\Sigma}(\boldsymbol{s}) = \begin{cases} \{\,(1,(\mathsf{C}{:}\mathtt{bag}))\,\} & \text{if } \boldsymbol{s} = \langle\rangle \\ \{\,(6,(\mathsf{C},\checkmark))\,\} & \text{if } \boldsymbol{s} = \boldsymbol{s_3} \\ \emptyset & \text{for any other play}\end{cases} \tag{2.1}$$

The strategy $\widetilde{\Sigma}$ for C in $\mathcal{G}_{\mathsf{Buyer}\,\|\,\mathsf{Seller}}$ is winning (actually it is the only winning strategy for the game of this example). Moreover, $\widetilde{\Sigma}$ is univocal (in our simple example there are not non univocal strategies for player C).

In a game $\mathcal{G}_{\rho\|\sigma}$ it is not restrictive to look, for player C, at univocal strategies only, as established in the next lemma.

We say that Σ *refines* Σ', written $\Sigma \leq \Sigma'$, if and only if $\Sigma(\boldsymbol{e}) \subseteq \Sigma'(\boldsymbol{e})$ for all $\boldsymbol{e}$.

Lemma 2.14. *If* C *has a winning strategy* Σ*, then* C *has a univocal winning strategy* Σ' *such that* $\Sigma' \leq \Sigma$*.*

Proof. Let $\Sigma(\boldsymbol{e}) = \{ e_1, \ldots, e_n \}$. Since events are a countable set we may suppose w.l.o.g. that there is a total and well founded ordering over E, so that if $\{ e_1, \ldots, e_n \} \neq \emptyset$ there exists an $e' = \min\{ e_1, \ldots, e_n \}$; let us define $\Sigma'(\boldsymbol{e}) = e'$.

By definition Σ' is a univocal refinement of Σ such that any play which conforms Σ' conforms Σ as well. Now if Σ is winning then the payoff $\Phi\mathsf{C}\boldsymbol{e} = 1$ for any play $\boldsymbol{e}$ conforming Σ and hence a fortiori for any play conforming Σ'; hence Σ' is winning for C. □

Remark 2.15. According to [7], A wins in a play if $\mathcal{W}A\boldsymbol{e} > 0$, where $\mathcal{W}A\boldsymbol{e} = \Phi A\boldsymbol{e}$ if all players are "innocent" in $\boldsymbol{e}$, while $\mathcal{W}A\boldsymbol{e} = -1$ if A is "culpable". Moreover, if A is innocent and someone else culpable, $\mathcal{W}A\boldsymbol{e} = +1$. A strategy Σ of A is winning if A wins in all *fair* plays conforming to Σ. A play $\boldsymbol{e}$ is "fair" for a strategy Σ of a player A if any event in E_A which is infinitely often enabled is eventually performed. Symmetrically A is "innocent" in $\boldsymbol{e}$ if she eventually plays all persistently enabled moves of her in $\boldsymbol{e}$, namely if she is fair to the other players, since the lack of a move by A might obstacle the moves by others; she is "culpable" otherwise. As said above, Definition 2.12 is a particularisation of the general definitions in [7]. In fact in a game $\mathcal{G}_{\rho\|\sigma}$ no move of any player can occur more than once in a play $\boldsymbol{e}$ because of time stamps. Therefore no move can be "persistently enabled", nor it can be prevented, since it can be enabled with a given time stamp only if there exists a legal transition in the LTS with the same label. Hence any player is innocent in a play $\boldsymbol{e}$ of $\mathcal{G}_{\rho\|\sigma}$ and all plays are fair. Therefore $\mathcal{W}$ coincides with Φ.

The last lemma is useful in proofs, and it is essentially a definition unfolding.

Lemma 2.16. *In a play* $\boldsymbol{e}$ *of a three-player game* $\mathcal{G}_{\rho\|\sigma}$*, player* A *wins if and only if either* $\boldsymbol{e}$ *is infinite or* $A = \mathsf{C}$ *and* $\boldsymbol{e}_{/k} = \langle e_0 e_1 \cdots (k, \mathsf{C}{:}\checkmark) \rangle$ *for some* k*.*

Proof. By Definition 2.12 $\Phi A\boldsymbol{e} > 0$ if either $\boldsymbol{e}$ is infinite or it contains a move $(k, (A : \checkmark))$ which has to be a move by A. But the only player that can play such a move is C (see Fig. 1). □

Notice that, by Remark 2.15, in our setting this lemma holds also in case we take into account the definition of *winning in a play* provided in [7], where $\mathcal{W}$ is used.

3. Retractable operational semantics and retractable compliance.

We provide now an operational semantics for client-server systems of affectible contracts, based on a rollback operation, as proposed in [4]. The rollback operation acts on the recording of the branches discarded during the *retractable* syncronizations. The retractable syncronizations are those involving affectible outputs. The actual sending of such outputs can hence be looked at as depending on an *agreement* between client and server. The computation can roll back to such agreement points when the client/server interaction gets stuck. When this happens, the interaction branch that had been followed up to this moment is discarded and another, possibly successful branch of interaction is pursued.

We begin by defining the notion of *contracts with history* as follows:

Definition 3.1 (Contracts with history)**.** Let $\mathsf{Histories}$ be the set of expressions (referred to also as *stacks*) generated by the grammar:

$$\boldsymbol{\gamma} ::= [\,] \mid \boldsymbol{\gamma} : \sigma \text{ where } \sigma \in \mathsf{ASC} \cup \{\circ\}.$$

Then the set of *contracts with history* is defined by:

$$\mathsf{RCH} = \{\boldsymbol{\gamma} \prec \sigma \mid \boldsymbol{\gamma} \in \mathsf{Histories}, \sigma \in \mathsf{ASC} \cup \{\circ\}\,\}.$$

Histories are finite lists of contracts representing the branches that have not been followed in the previous agreement points. The effect of rolling back to the last agreement point is modeled by restoring the last contract of the history as the current contract and by trying a different branch, if any. In case a contract in a history was a sum whose branches have all been tried, the empty sum of remaining alternatives is represented by $\circ$ (see the next definition of transition). We use the notation $\boldsymbol{\gamma} \prec \sigma$ for the contract σ with history γ.

This is formalised by the operational semantics of contracts with histories that is defined as follows.

Definition 3.2 (LTS of Contracts with History)**.**

$$(+)\quad \boldsymbol{\gamma} \prec \alpha.\sigma + \sigma' \xrightarrow{\alpha} \boldsymbol{\gamma} : \sigma' \prec \sigma \qquad (\oplus)\quad \boldsymbol{\gamma} \prec \overline{a}.\sigma \oplus \sigma' \xrightarrow{\tau} \boldsymbol{\gamma} \prec \overline{a}.\sigma$$

$$(\alpha)\quad \boldsymbol{\gamma} \prec \alpha.\sigma \xrightarrow{\alpha} \boldsymbol{\gamma} : \circ \prec \sigma \qquad (\mathsf{rb})\quad \boldsymbol{\gamma} : \sigma' \prec \sigma \xrightarrow{\mathsf{rb}} \boldsymbol{\gamma} \prec \sigma'$$

Rule $(+)$ formalises the selection of a branch of an external choice. The effect of the selection is that the unselected branches are memorised on top of the stack as last contract of the history. Notice that the memorization on the stack occurs also in case of an external input choice, like for instance in $\boldsymbol{\gamma} \prec a.\sigma_1 + b.\sigma_2 \xrightarrow{a} \boldsymbol{\gamma} : b.\sigma_2 \prec \sigma_1$. This because a possible partner could be an external output choice and the resulting syncronization would hence be a retractable one. However, in case the partner were an internal choice and hence in case the resulting syncronization were unretractable, the memorized contract should not be taken into account by any future rollback. The operational semantics manages to take care of such a possibility by means of the use of the symbol '$\circ$' (see Example 3.4 below).
Rule $(\oplus)$ formalises the fact that when an internal choice occurs, the stack remains unchanged. Rule (α) concerns instead the execution of a single action: the history is modified by pushing a '$\circ$' on the stack, meaning that the only available branch has been tried and no alternative is left. Rule (rb) recovers the contract on the top of the stack (if the stack is different from $[\,]$) and replaces the current one with it. Note that the combined effect of rules $(\oplus)$ and (α) is that the alternative branches of an internal choice are unrecoverable.

The interaction of a client with a server is modeled by the reduction of their parallel composition, that can be either forward, consisting of CCS style synchronisations and single internal choices, or backward if there is no possible forward reduction, the client is different than $\mathbf{1}$ (the fulfilled contract) and rule (rb) is applicable on both sides.

Definition 3.3 (TS of Client/Server Pairs)**.** We define the relation $\longrightarrow$ over pairs of contracts with histories by the following rules:

$$\frac{\boldsymbol{\delta} \prec \rho \xrightarrow{\alpha} \boldsymbol{\delta'} \prec \rho' \qquad \boldsymbol{\gamma} \prec \sigma \xrightarrow{\overline{\alpha}} \boldsymbol{\gamma'} \prec \sigma'}{\boldsymbol{\delta} \prec \rho \parallel \boldsymbol{\gamma} \prec \sigma \longrightarrow \boldsymbol{\delta'} \prec \rho' \parallel \boldsymbol{\gamma'} \prec \sigma'} \ (\mathit{comm})$$

$$\frac{\boldsymbol{\delta} \prec \rho \xrightarrow{\tau} \boldsymbol{\delta} \prec \rho'}{\boldsymbol{\delta} \prec \rho \parallel \boldsymbol{\gamma} \prec \sigma \longrightarrow \boldsymbol{\delta} \prec \rho' \parallel \boldsymbol{\gamma} \prec \sigma} \ (\tau)$$

$$\frac{\boldsymbol{\gamma} \prec \rho \xrightarrow{\mathsf{rb}} \boldsymbol{\gamma'} \prec \rho' \qquad \boldsymbol{\delta} \prec \sigma \xrightarrow{\mathsf{rb}} \boldsymbol{\delta'} \prec \sigma' \qquad \rho \neq \mathbf{1}}{\boldsymbol{\gamma} \prec \rho \parallel \boldsymbol{\delta} \prec \sigma \longrightarrow \boldsymbol{\gamma'} \prec \rho' \parallel \boldsymbol{\delta'} \prec \sigma'} \ (\mathit{rbk})$$

plus the rule symmetric to (τ) w.r.t. $\parallel$. Moreover, rule (*rbk*) applies only if neither (*comm*) nor (τ) do.

Example 3.4. Let us consider the following client/server system

$$\boldsymbol{\gamma} \prec a.c.\rho_1 + b.\rho_2 \parallel \boldsymbol{\delta} \prec \overline{a}.d.\sigma_1 + \overline{b}.\sigma_2$$

We are in presence of an agreement point, and the operational semantics defined above is such that during the following syncronization the unchosen branches are memorized:

$$\boldsymbol{\gamma} \prec a.c.\rho_1 + b.\rho_2 \parallel \boldsymbol{\delta} \prec \overline{a}.d.\sigma_1 + \overline{b}.\sigma_2 \longrightarrow \boldsymbol{\gamma} : b.\rho_2 \prec c.\rho_1 \parallel \boldsymbol{\delta} : \overline{b}.\sigma_2 \prec d.\sigma_1$$

Now, no syncronization is possible in the system $\boldsymbol{\gamma} : b.\rho_2 \prec c.\rho_1 \parallel \boldsymbol{\delta} : \overline{b}.\sigma_2 \prec d.\sigma_1$, and hence the rules dealing with rollback allow to recover the branches which were not selected in the previous agreement point:

$$\boldsymbol{\gamma} : b.\rho_2 \prec c.\rho_1 \parallel \boldsymbol{\delta} : \overline{b}.\sigma_2 \prec d.\sigma_1 \longrightarrow \boldsymbol{\gamma} \prec b.\rho_2 \parallel \boldsymbol{\delta} \prec \overline{b}.\sigma_2$$

The interaction can now proceed along the previously unchosen branches.

Let us consider now, instead, the following client/server system:

$$\boldsymbol{\gamma} \prec a.c.\rho_1 + b.\rho_2 \parallel \boldsymbol{\delta} \prec \overline{a}.d.\sigma_1 \oplus \overline{b}.\sigma_2$$

We are not in presence of an agreement point and the interaction involves an internal choice by the server:

$$\boldsymbol{\gamma} \prec a.c.\rho_1 + b.\rho_2 \parallel \boldsymbol{\delta} \prec \overline{a}.d.\sigma_1 \oplus \overline{b}.\sigma_2 \longrightarrow \boldsymbol{\gamma} \prec a.c.\rho_1 + b.\rho_2 \parallel \boldsymbol{\delta} \prec \overline{a}.d.\sigma_1$$

Now the syncronization in $\boldsymbol{\gamma} \prec a.c.\rho_1 + b.\rho_2 \parallel \boldsymbol{\delta} \prec \overline{a}.d.\sigma_1$ causes, in the client, the memorization of $b.\rho_2$, namely the client's branch not selected by the output $\overline{a}$. In the server, instead, the single action $\overline{a}$ causes the memorization of '$\circ$' on the stack:

$$\boldsymbol{\gamma} \prec a.c.\rho_1 + b.\rho_2 \parallel \boldsymbol{\delta} \prec \overline{a}.d.\sigma_1 \longrightarrow \boldsymbol{\gamma} : b.\rho_2 \prec c.\rho_1 \parallel \boldsymbol{\delta} : \circ \prec d.\sigma_1$$

Since the system $\boldsymbol{\gamma} : b.\rho_2 \prec c.\rho_1 \parallel \boldsymbol{\delta} : \circ \prec d.\sigma_1$ is stuck, we recover the tops of the stacks as current contracts:

$$\boldsymbol{\gamma} : b.\rho_2 \prec c.\rho_1 \parallel \boldsymbol{\delta} : \circ \prec d.\sigma_1 \longrightarrow \boldsymbol{\gamma} \prec b.\rho_2 \parallel \boldsymbol{\delta} \prec \circ$$

The fact that '$\circ$' cannot synchronize with anything forces us to apply again rule (rbk) on $\boldsymbol{\gamma} \prec b.\rho_2 \parallel \boldsymbol{\delta} \prec \circ$, that is to keep on rolling back in order to get to the first agreement point memorized in $\boldsymbol{\gamma}$ and $\boldsymbol{\delta}$. The contract $b.\rho_2$ and the symbol '$\circ$' recovered from the top of the stacks will be simply "thrown away".

Up to the rollback mechanism, compliance in the retractable setting is defined as usual with client-server contracts.

Definition 3.5 (Retractable Compliance, $\dashv^{\mathsf{rdk}}$)**.**

(1) The binary relation $\dashv^{\mathsf{rdk}}$ on contracts with histories is defined as follows:
for any $\boldsymbol{\delta'}, \rho', \boldsymbol{\gamma'}, \sigma'$, $\boldsymbol{\delta} \prec \rho \dashv^{\mathsf{rdk}} \boldsymbol{\gamma} \prec \sigma$ holds whenever

$$\boldsymbol{\delta} \prec \rho \parallel \boldsymbol{\gamma} \prec \sigma \xrightarrow{*} \boldsymbol{\delta'} \prec \rho' \parallel \boldsymbol{\gamma'} \prec \sigma' \not\longrightarrow \text{ implies } \rho' = \mathbf{1}$$

(2) The relation $\dashv^{\mathsf{rdk}}$ on affectible contracts is defined by: $\rho \dashv^{\mathsf{rdk}} \sigma$ if $[\,] \prec \rho \dashv^{\mathsf{rdk}} [\,] \prec \sigma$.

Example 3.6 ([4])**.** In the Buyer/Seller example we have that, in case a belt is agreed upon and the buyer decides to pay using her credit card, the system gets stuck in an unsuccessful state. This causes a rollback enabling a successful state to be reached.

$$\begin{array}{lrcl}
 & [\,] \prec \mathsf{Buyer} & \parallel & [\,] \prec \mathsf{Seller} \\
\xrightarrow{\mathsf{comm}} & \overline{\mathtt{bag}}.\mathtt{price}.(\overline{\mathtt{card}} \oplus \overline{\mathtt{cash}}) \prec \mathtt{price}.(\overline{\mathtt{card}} \oplus \overline{\mathtt{cash}}) & \parallel & \mathtt{bag}.\overline{\mathtt{price}}.(\mathtt{card} + \mathtt{cash}) \prec \overline{\mathtt{price}}.\mathtt{cash} \\
\xrightarrow{\mathsf{comm}} & \overline{\mathtt{bag}}.\mathtt{price}.(\overline{\mathtt{card}} \oplus \overline{\mathtt{cash}}) : \circ \prec (\overline{\mathtt{card}} \oplus \overline{\mathtt{cash}}) & \parallel & \mathtt{bag}.\overline{\mathtt{price}}.(\mathtt{card} + \mathtt{cash}) : \circ \prec \mathtt{cash} \\
\xrightarrow{\tau} & \overline{\mathtt{bag}}.\mathtt{price}.(\overline{\mathtt{card}} \oplus \overline{\mathtt{cash}}) : \circ \prec \overline{\mathtt{card}} & \parallel & \mathtt{bag}.\overline{\mathtt{price}}.(\mathtt{card} + \mathtt{cash}) : \circ \prec \mathtt{cash} \\
\xrightarrow{\mathsf{rbk}} & \overline{\mathtt{bag}}.\mathtt{price}.(\overline{\mathtt{card}} \oplus \overline{\mathtt{cash}}) \prec \circ & \parallel & \mathtt{bag}.\overline{\mathtt{price}}.(\mathtt{card} + \mathtt{cash}) \prec \circ \\
\xrightarrow{\mathsf{rbk}} & [\,] \prec \overline{\mathtt{bag}}.\mathtt{price}.(\overline{\mathtt{card}} \oplus \overline{\mathtt{cash}}) & \parallel & [\,] \prec \mathtt{bag}.\overline{\mathtt{price}}.(\mathtt{card} + \mathtt{cash}) \\
\xrightarrow{\mathsf{comm}} & \circ \prec \mathtt{price}.(\overline{\mathtt{card}} \oplus \overline{\mathtt{cash}}) & \parallel & \circ \prec \overline{\mathtt{price}}.(\mathtt{card} + \mathtt{cash}) \\
\xrightarrow{\mathsf{comm}} & \circ : \circ \prec (\overline{\mathtt{card}} \oplus \overline{\mathtt{cash}}) & \parallel & \circ : \circ \prec (\mathtt{card} + \mathtt{cash}) \\
\xrightarrow{\tau} & \circ : \circ \prec \overline{\mathtt{card}} & \parallel & \circ : \circ \prec (\mathtt{card} + \mathtt{cash}) \\
\xrightarrow{\mathsf{comm}} & \circ : \circ : \circ \prec \mathbf{1} & \parallel & \circ : \circ : \mathtt{cash} \prec \mathbf{1} \\
\not\longrightarrow & & &
\end{array}$$

There are other possible reduction sequences: the one in which a belt is agreed upon but the buyer decides to pay by cash; the one in which a bag is agreed upon and the buyer decides to pay by cash; the one in which a bag is agreed upon and the buyer decides to pay by credit card.
It is possible to check that also for all these other reduction sequences a successful state is reached. So we have that $\mathsf{Buyer} \dashv^{\mathsf{rdk}} \mathsf{Seller}$.

It can be checked that if $\boldsymbol{\gamma} \prec \rho \parallel \boldsymbol{\delta} \prec \sigma \longrightarrow \boldsymbol{\gamma'} \prec \rho' \parallel \boldsymbol{\delta'} \prec \sigma'$ and $\boldsymbol{\gamma}$ and $\boldsymbol{\delta}$ have the same length, then this is also true of $\boldsymbol{\gamma'}$ and $\boldsymbol{\delta'}$.
Now, let $[\,] \prec \rho \parallel [\,] \prec \sigma \xrightarrow{*} \boldsymbol{\gamma'} \prec \rho' \parallel \boldsymbol{\delta'} \prec \sigma' \not\longrightarrow$. Since $\boldsymbol{\gamma'} \prec \rho' \parallel \boldsymbol{\delta'} \prec \sigma' \not\longrightarrow$, a fortiori rule (*rbk*) does not apply and then, by rule (rb), either $\boldsymbol{\gamma'}$ or $\boldsymbol{\delta'}$ must be empty. Besides, by what said before, they must have the same lenght. Hence they are both empty.
This is stated in the item (1) of the following lemma (see Appendix B.2 for the proof) and it will be essential, together with item (2), to prove some relevant results in Section 5 about the retractable compliance relation.

Lemma 3.7.

(1) *If* $[\,] \prec \rho \parallel [\,] \prec \sigma \xrightarrow{*} \boldsymbol{\delta} \prec \rho' \parallel \boldsymbol{\gamma} \prec \sigma' \not\longrightarrow$, *then* $\boldsymbol{\delta} = \boldsymbol{\gamma} = [\,]$.
(2) *If* $\boldsymbol{\delta} \prec \rho \dashv^{\mathsf{rdk}} \boldsymbol{\gamma} \prec \sigma$, *then* $\boldsymbol{\delta'} : \boldsymbol{\delta} \prec \rho \dashv^{\mathsf{rdk}} \boldsymbol{\gamma'} : \boldsymbol{\gamma} \prec \sigma$ *for all* $\boldsymbol{\delta'}$, $\boldsymbol{\gamma'}$.

4. Orchestrated operational semantics and orchestrated compliance

In the present section we define an operational semantics for affectible contracts and for client-server systems, where the interaction between a client and a server – in particular when affectible outputs are concerned – is driven by a third process, a mediator, called *orchestrator*. For a general discussion on orchestrators for contracts and session-contracts, we refer to [16, 15] and [6] respectively.

Definition 4.1 (LTS for affectible session contracts)**.** Let $\mathbf{Act} = \mathcal{N} \cup \overline{\mathcal{N}} \cup \{\, \overline{a}^{+} \mid \overline{a} \in \overline{\mathcal{N}} \,\}$.

$$(+)\quad a.\sigma + \sigma' \xrightarrow{a} \sigma \qquad (\overline{+})\quad \overline{a}.\sigma + \sigma' \xrightarrow{\overline{a}^{+}} \sigma$$

$$(\oplus)\quad \overline{a}.\sigma \oplus \sigma' \longrightarrow \overline{a}.\sigma \qquad (\alpha)\quad \alpha.\sigma \xrightarrow{\alpha} \sigma$$

As for other orchestrated formalisms, the task of an orchestrator is to mediate the interaction between a client and a server, by selecting or constraining the possible interactions. In the present setting orchestrators, that we dub *strategy-orchestrators*, are defined as a variant of the session-orchestrators of [6], which in turn are a restriction of orchestrators in [16].

A strategy orchestrator does mediate between two affectible session contracts by selecting one of the possible affectible choices and constraining non-affectible ones. The orchestration actions have two possible shapes: either $\langle \alpha, \overline{\alpha} \rangle$, enabling the unaffectible synchronization, or $\langle \alpha, \overline{\alpha} \rangle^{+}$, enabling the affectible synchronization.

Definition 4.2 (Strategy Orchestrators)**.**

(1) The set **OrchAct** of *strategy-orchestration actions* is defined by

$$\mathbf{OrchAct} = \{\, \langle \alpha, \overline{\alpha} \rangle \mid \alpha \in \mathcal{N} \cup \overline{\mathcal{N}} \,\} \cup \{\, \langle \alpha, \overline{\alpha} \rangle^{+} \mid \alpha \in \mathcal{N} \cup \overline{\mathcal{N}} \,\}$$

We let $\mu, \mu', \ldots$ range over elements of **OrchAct** with the shape $\langle \alpha, \overline{\alpha} \rangle$, and $\mu^{+}, \mu'^{+}, \ldots$ range over elements of **OrchAct** with the shape $\langle \alpha, \overline{\alpha} \rangle^{+}$.

(2) We define the set Orch of *strategy orchestrators*, ranged over by $f, g, \ldots$, as the *closed* (with respect to the binder rec) terms generated by the following grammar:

$$\begin{array}{rcll} f, g & ::= & \mathtt{1} & \text{(idle)} \\ & \mid & \mu^{+}.f & \text{(prefix)} \\ & \mid & \mu_1.f_1 \vee \ldots \vee \mu_n.f_n & \text{(disjunction)} \\ & \mid & x & \text{(variable)} \\ & \mid & \mathsf{rec}\, x.f & \text{(recursion)} \end{array}$$

where the μ_i's in a disjunction are pairwise distinct. Moreover, we impose strategy orchestrators to be *contractive*, i.e. the f in $\mathsf{rec}\, x.f$ is assumed not to be a variable.

We write $\bigvee_{i \in I} \mu_i.f_i$ as short for $\mu_1.f_1 \vee \ldots \vee \mu_n.f_n$, where $I = \{\, 1, \ldots, n \,\}$.
If not stated otherwise, we consider recursive orchestrators up-to unfolding, that is we equate $\mathsf{rec}\, x.f$ with $f\{x/\mathsf{rec}\, x.f\}$. We omit trailing $\mathtt{1}$'s.

Strategy orchestrators are "simple orchestrators" in [16] and "synchronous orchestrators" in [15], but for the kind of prefixes which are allowed in a single prefix or in a disjunction. In fact a prefix $\langle \alpha, \overline{\alpha} \rangle^{+}$ cannot occur in disjunctions, where all the orchestrators must be prefixed by $\langle \alpha, \overline{\alpha} \rangle$ actions. When clear from the context we shall refer to stategy orchestrators just as orchestrators.

Definition 4.3 (Strategy orchestrators LTS)**.** We define the labelled transition system $(\mathsf{Orch}, \mathbf{OrchAct}, \mapsto)$ by

$$\mu^+.f \overset{\mu^+}{\mapsto} f \qquad\qquad (\bigvee_{i\in I} \mu_i.f_i) \overset{\mu_k}{\mapsto} f_k \quad (k \in I)$$

An *orchestrated system* $\rho \parallel_f \sigma$ is the client-server system $\rho \parallel \sigma$ whose interactions are mediated by the orchestrator f.

Definition 4.4 (LTS for orchestrated systems[4])**.**
Let $\rho, \sigma \in \mathsf{ASC}$ and $f \in \mathsf{Orch}$.

$$\frac{\rho \to \rho'}{\rho \parallel_f \sigma \to \rho' \parallel_f \sigma} \qquad \frac{\sigma \to \sigma'}{\rho \parallel_f \sigma \to \rho \parallel_f \sigma'}$$

$$\frac{\rho \xrightarrow{a} \rho' \quad f \overset{\langle \overline{a}, a\rangle^+}{\mapsto} f' \quad \sigma \xrightarrow{\overline{a}^+} \sigma'}{\rho \parallel_f \sigma \xrightarrow{+} \rho' \parallel_{f'} \sigma'} \qquad \frac{\rho \xrightarrow{\overline{a}^+} \rho' \quad f \overset{\langle a, \overline{a}\rangle^+}{\mapsto} f' \quad \sigma \xrightarrow{a} \sigma'}{\rho \parallel_f \sigma \xrightarrow{+} \rho' \parallel_{f'} \sigma'}$$

$$\frac{\rho \xrightarrow{\alpha} \rho' \quad f \overset{\langle \overline{\alpha}, \alpha\rangle}{\mapsto} f' \quad \sigma \xrightarrow{\overline{\alpha}} \sigma'}{\rho \parallel_f \sigma \xrightarrow{\tau} \rho' \parallel_f \sigma'} \ (\alpha \in \mathcal{N} \cup \overline{\mathcal{N}})$$

Moreover we define $\Longrightarrow = \to^* \circ (\xrightarrow{\tau} \cup \xrightarrow{+})$.

In both transitions $\xrightarrow{+}$ and $\xrightarrow{\tau}$, synchronization may happen only if the orchestrator has a transition with the appropriate orchestration action. This is because in an orchestrated interaction both client and server are committed to the synchronizations allowed by the orchestrator only. Because of its structure an orchestrator always selects exactly one synchronisation of affectible actions on client and server side, while the disjunction of orchestrators represents the constraint that only certain synchronisations of unaffectible actions are permitted.

Definition 4.5 (Strategy-orchestrated Compliance)**.**
(1) $f : \rho \dashv^{\mathsf{Orch}} \sigma$ if for any ρ' and σ', the following holds:

$$\rho \parallel_f \sigma \Longrightarrow^* \rho' \parallel_{f'} \sigma' \not\Longrightarrow \quad \text{implies} \quad \rho' = \mathbf{1}.$$

(2) $\rho \dashv^{\mathsf{Orch}} \sigma$ if $\exists f.\, [\, f : \rho \dashv^{\mathsf{Orch}} \sigma\,]$.

Example 4.6. Let $f = \langle \mathtt{bag}, \overline{\mathtt{bag}}\rangle^+.\langle \overline{\mathtt{price}}, \mathtt{price}\rangle.(\langle \mathtt{card}, \overline{\mathtt{card}}\rangle.\mathtt{1} \vee \langle \mathtt{cash}, \overline{\mathtt{cash}}\rangle.\mathtt{1})$. For our Buyer/Seller example, we can notice that the orchestrator f forces the buyer and the seller to agree on a bag, and then leaves to the client the choice of the type of payment. Hence, in case the client chooses to pay by card, the orchestrated client/server interaction proceeds as follows.

[4]The present LTS corresponds to a three-way synchronous communication, which could cause some problems from the implementation point of view. We abstract away from these problems in the theoretical setting of the present paper; an actual implementation might modularize each orchestrated synchronizations into two distinct binary synchronizations.

$$(\textsc{Ax}) : \overline{\Gamma \rhd \mathbf{1} \backsim \sigma} \qquad (\textsc{Hyp}) : \overline{\Gamma, \rho \backsim \sigma \rhd \rho \backsim \sigma}$$

$$(+\cdot+) : \frac{\Gamma, \alpha.\rho + \rho' \backsim \overline{\alpha}.\sigma + \sigma' \rhd \rho \backsim \sigma}{\Gamma \rhd \alpha.\rho + \rho' \backsim \overline{\alpha}.\sigma + \sigma'}$$

$$(\oplus\cdot+) : \frac{\forall i \in I.\ \Gamma, \bigoplus_{i\in I}\overline{a}_i.\rho_i \backsim \sum_{j\in I\cup J} a_j.\sigma_j \rhd \rho_i \backsim \sigma_i}{\Gamma \rhd \bigoplus_{i\in I}\overline{a}_i.\rho_i \backsim \sum_{j\in I\cup J} a_j.\sigma_j}$$

$$(+\cdot\oplus) : \frac{\forall i \in I.\ \Gamma, \sum_{j\in I\cup J} a_j.\sigma_j \backsim \bigoplus_{i\in I}\overline{a}_i.\rho_i \rhd \rho_i \backsim \sigma_i}{\Gamma \rhd \sum_{j\in I\cup J} a_j.\sigma_j \backsim \bigoplus_{i\in I}\overline{a}_i.\rho_i}$$

Figure 2: System $\rhd$

		Buyer	$\|_{\langle \mathtt{bag},\overline{\mathtt{bag}}\rangle^+.\langle\overline{\mathtt{price}},\mathtt{price}\rangle.(\langle\mathtt{card},\overline{\mathtt{card}}\rangle.\mathtt{1}\vee\langle\mathtt{cash},\overline{\mathtt{cash}}\rangle.\mathtt{1})}$	Seller
$\longrightarrow$	$\mathtt{price}.(\overline{\mathtt{card}} \oplus \overline{\mathtt{cash}})$		$\|_{\langle\overline{\mathtt{price}},\mathtt{price}\rangle.(\langle\mathtt{card},\overline{\mathtt{card}}\rangle.\mathtt{1}\vee\langle\mathtt{cash},\overline{\mathtt{cash}}\rangle.\mathtt{1})}$	$\overline{\mathtt{price}}.(\mathtt{card} + \mathtt{cash})$
$\longrightarrow$	$(\overline{\mathtt{card}} \oplus \overline{\mathtt{cash}})$		$\|_{\langle\mathtt{card},\overline{\mathtt{card}}\rangle.\mathtt{1}\vee\langle\mathtt{cash},\overline{\mathtt{cash}}\rangle.\mathtt{1}}$	$\mathtt{card} + \mathtt{cash}$
$\longrightarrow$	$\overline{\mathtt{card}}$		$\|_{\langle\mathtt{card},\overline{\mathtt{card}}\rangle.\mathtt{1}\vee\langle\mathtt{cash},\overline{\mathtt{cash}}\rangle.\mathtt{1}}$	$\mathtt{card} + \mathtt{cash}$
$\longrightarrow$	$\mathbf{1}$		$\|_{\mathtt{1}}$	$\mathbf{1}$
$\not\longrightarrow$				

Since there are not infinite reduction sequences and a succesful state is reached for any other possible maximal reduction sequence (the only other possible one being the one where the buyer chooses to pay by cash), we have that $f : \mathsf{Buyer} \dashv^{\mathsf{Orch}} \mathsf{Seller}$.

Remark 4.7. As we shall see, strategy-orchestrators do correspond to univocal strategies for player C in $\mathcal{G}_{\rho\|\sigma}$ and are technically easier to work with. (In the proofs it will be easier to deal with partial functions rather than relations). On the other hand, we can recover a full correspondence among unrestricted strategies for C and orchestrators by allowing disjunctions of affectible synchronization actions $\langle\alpha,\overline{\alpha}\rangle^+$. In a session-based scenario, however, we expect any nondeterminism to depend solely on either the client or the server. By allowing $f = \langle\overline{a}, a\rangle^+.f_1 \vee \langle\overline{b}, b\rangle^+.f_2$ in the system $a.\rho_1 + b.\rho_2 \|_f \overline{a}.\sigma_1 + \overline{b}.\sigma_2$, the nondeterminism would depend on the orchestrator as well.

5. Linking up all together: Main Results.

In this section we connect all the notions defined up to now. We begin by showing that the relation $\mathbb{AC}$ can be axiomatized by means of the following formal system. In such a system the symbol $\backsim$ is the syntactic counterpart for the $\mathbb{AC}$ relation.

Definition 5.1 (Formal system $\rhd$ for $\mathbb{AC}$)**.** An environment Γ is a finite set of expressions of the form $\delta \backsim \gamma$ where $\delta, \gamma \in \mathsf{ASC}$. The judgments of System $\rhd$ are expressions of the form $\Gamma \rhd \rho \backsim \sigma$. The axioms and rules of $\rhd$ are as in Figure 5, where in rule $(+\cdot+)$ we assume that a term of the form $a.\rho$ can be used instead of $a.\rho + \rho'$.

We shall write $\rhd\ \rho \backsim \sigma$ as an abbreviation for $\emptyset \rhd \rho \backsim \sigma$. Moreover, we shall write $\mathcal{D} :: \Gamma \rhd \rho \backsim \sigma$ to denote that $\mathcal{D}$ is a derivation having the judgment $\Gamma \rhd \rho \backsim \sigma$ as conclusion. When clear from the context, we shall write simply $\Gamma \rhd \rho \backsim \sigma$ to state that there exists a derivation $\mathcal{D}$ such that $\mathcal{D} :: \Gamma \rhd \rho \backsim \sigma$.

Example 5.2. Let us derive in system $\rhd$ the judgment $\emptyset \rhd \mathsf{Buyer} \backsim \mathsf{Seller}$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma'' \rhd \mathbf{1} \backsim \mathbf{1}}\,(\mathrm{Ax}) \qquad \dfrac{}{\Gamma'' \rhd \mathbf{1} \backsim \mathbf{1}}\,(\mathrm{Ax})}{\Gamma' \rhd \overline{\mathtt{card}} \oplus \overline{\mathtt{cash}} \backsim \mathtt{card} + \mathtt{cash}}\,(\oplus, +)}{\mathsf{Buyer} \backsim \mathsf{Seller} \rhd \mathtt{price}.(\overline{\mathtt{card}} \oplus \overline{\mathtt{cash}}) \backsim \overline{\mathtt{price}}.(\mathtt{card} + \mathtt{cash})}\,(+, +)}{\rhd \mathsf{Buyer} \backsim \mathsf{Seller}}\,(+, +) \tag{5.1}$$

where

$$\Gamma' = \mathsf{Buyer} \backsim \mathsf{Seller},\ \mathtt{price}.(\overline{\mathtt{card}} \oplus \overline{\mathtt{cash}}) \backsim \overline{\mathtt{price}}.(\mathtt{card} + \mathtt{cash})$$
$$\text{and } \Gamma'' = \Gamma',\ \overline{\mathtt{card}} \oplus \overline{\mathtt{cash}} \backsim \mathtt{card} + \mathtt{cash}$$

We now prove the relevant properties of this system. First, the proposed axiomatisation exactly captures $\mathbb{AC}$ as intended.

5.1. **Soundness and Completeness of $\rhd$ with respect to $\mathbb{AC}$.** In this subsection we prove that the system $\rhd$ is sound and complete with respect to the affectible compliance relation $\mathbb{AC}$, namely

$$\rhd \rho \backsim \sigma \quad \textit{if and only if} \quad \rho\, \mathbb{AC}\, \sigma$$

Moreover, we shall get decidability of $\mathbb{AC}$ as a corollary.

We begin by showing that the relation $\mathbb{AC}$ of Definition 1.2 can be equivalently defined in a stratified way.

Definition 5.3.

(1) Let $\mathcal{K} : \mathcal{P}(\mathsf{ASC} \times \mathsf{ASC}) \to \mathcal{P}(\mathsf{ASC} \times \mathsf{ASC})$ be such that, for any $\mathcal{R} \subseteq \mathsf{ASC} \times \mathsf{ASC}$, we get $(\rho, \sigma) \in \mathcal{K}(\mathcal{R})$ if either $\rho = \mathbf{1}$ or one of the following holds:
   (a) $\rho = \sum_{i \in I} \alpha_i.\rho_i$, $\sigma = \sum_{j \in J} \overline{\alpha}_j.\sigma_j$ and $\exists h \in I \cap J.\ (\rho_h, \sigma_h) \in \mathcal{R}$;
   (b) $\rho = \bigoplus_{i \in I} \overline{a}_i.\rho_i$, $\sigma = \sum_{j \in J} a_j.\sigma_j$, $I \subseteq J$ and $\forall h \in I.\ (\rho_h, \sigma_h) \in \mathcal{R}$;
   (c) $\rho = \sum_{j \in J} a_j.\rho_j$, $\sigma = \bigoplus_{i \in I} \overline{a}_i.\sigma_i$, $I \subseteq J$ and $\forall h \in I.\ (\rho_h, \sigma_h) \in \mathcal{R}$.

(2) A relation $\mathcal{R} \subseteq \mathsf{ASC} \times \mathsf{ASC}$ is an *affectible compliance relation* if $\mathcal{R} \subseteq \mathcal{K}(\mathcal{R})$.

(3) For any $n \in \mathbb{N}$ we define $\mathbb{AC}_n \subseteq \mathsf{ASC} \times \mathsf{ASC}$ as follows:
$\mathbb{AC}_0 = \mathsf{ASC} \times \mathsf{ASC}$, whereas, for $n > 0$ $\mathbb{AC}_n = \mathcal{K}(\mathbb{AC}_{n-1})$

(4) We define $\mathbb{AC}_{co} = \bigcap_{n \in \mathbb{N}} \mathbb{AC}_n$

Fact 5.4. $\mathbb{AC} = \mathbb{AC}_{co} = \nu(\mathcal{K})$.

Notice that $\mathbb{AC}_k \subseteq \mathbb{AC}_{k-1}$ for all k. We define now a stratified notion of validity for judgments in system $\rhd$.

Definition 5.5 (Stratified $\mathbb{AC}$-semantics for $\rhd$)**.** Let Γ be a set of statements of the form $\rho \backsim \sigma$ and let $k \in \mathbb{N}$. We define

(1) $\models_k^{\mathbb{AC}} \Gamma$ if $\forall (\rho' \backsim \sigma') \in \Gamma.\ \rho'\ \mathbb{AC}_k\, \sigma'$;

(2) $\Gamma \models_k^{\mathbb{AC}} \rho \backsim \sigma$ if $\models_k^{\mathbb{AC}} \Gamma \Rightarrow \rho\ \mathbb{AC}_k\, \sigma$.

We can now proceed by proving the soundness of $\rhd$.

Prove$(\Gamma \rhd \rho \multimap \sigma)$

if $\rho = \mathbf{1}$ **then** $(\textsc{Ax}) : \dfrac{}{\Gamma \rhd \mathbf{1} \multimap \sigma}$

else if $\rho \multimap \sigma \in \Gamma$ **then** $(\textsc{Hyp}) : \dfrac{}{\Gamma, \rho \multimap \sigma \rhd \rho \multimap \sigma}$

else if $\rho = \bigoplus_{i\in I} \overline{a}_i.\rho_i$ **and** $\sigma = \sum_{j\in J} a_j.\sigma_j$ **and** $I \subseteq J$
and for all $k \in I$ $\mathcal{D}_k = \mathbf{Prove}(\Gamma, \rho \multimap \sigma \rhd \rho_k \multimap \sigma_k) \neq \mathbf{fail}$
then $(\oplus \cdot +) : \dfrac{(\forall k \in I)\ \mathcal{D}_k}{\Gamma \rhd \rho \multimap \sigma}$

else if $\rho = \sum_{i\in I} a_i.\rho_i$ **and** $\sigma = \bigoplus_{j\in J} \overline{a}_j.\sigma_j$ **and** $I \supseteq J$
and for all $k \in J$ $\mathcal{D}_k = \mathbf{Prove}(\Gamma, \rho \multimap \sigma \rhd \rho_k \multimap \sigma_k) \neq \mathbf{fail}$
then $(+ \cdot \oplus) : \dfrac{(\forall k \in J)\ \mathcal{D}_k}{\Gamma \rhd \rho \multimap \sigma}$ **else fail**

else if $\rho = \sum_{i\in I} \alpha_i.\rho_i$ **and** $\sigma = \sum_{j\in J} \overline{\alpha}_j.\sigma_j$
and exists $k \in I \cap J$ **s.t.** $\mathcal{D} = \mathbf{Prove}(\Gamma, \rho \multimap \sigma \rhd \rho_k \multimap \sigma_k) \neq \mathbf{fail}$
then $(+ \cdot +) : \dfrac{\mathcal{D}}{\Gamma \rhd \rho \multimap \sigma}$ **else fail**

else fail

Figure 3: The procedure **Prove**.

Proposition 5.6 (Soundness of $\rhd$)**.** *If* $\rhd \rho \multimap \sigma$*, then* $\models^{\mathbb{AC}} \rho \multimap \sigma$*.*

Proof. Actually we show a stronger property, namely that

$$\Gamma \rhd \rho \multimap \sigma \text{ implies } \Gamma \models^{\mathbb{AC}} \rho \multimap \sigma.$$

By Fact 5.4, it is enough to show that

$$\Gamma \rhd \rho \multimap \sigma \quad \text{implies} \quad \Gamma \models^{\mathbb{AC}}_k \rho \multimap \sigma \text{ for all } k.$$

We proceed by simultaneous induction over the derivation $\mathcal{D} :: \Gamma \rhd \rho \multimap \sigma$ and over k. Since $\Gamma \models^{\mathbb{AC}}_0 \rho \multimap \sigma$ trivially holds, we shall keep the case $k = 0$ implicit. Let $k > 0$; we distinguish the possible cases of the last rule in $\mathcal{D}$.

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

$$\frac{}{\Gamma \rhd \mathbf{1} \multimap \sigma}\ (\textsc{Ax})$$

and the thesis is immediate since $\mathbf{1}\ \mathbb{AC}_k\ \sigma$;

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

$$\frac{}{\Gamma, \rho \multimap \sigma \rhd \rho \multimap \sigma}\ (\textsc{Hyp})$$

and $\Gamma, \rho \multimap \sigma \models^{\mathbb{AC}}_k \rho \multimap \sigma$ trivially holds.

Case $(+\cdot+)$**:** Then $\mathcal{D}$ ends by

$$\frac{\Gamma, \alpha.\rho+\rho' \backsim \overline{\alpha}.\sigma+\sigma' \rhd \rho \backsim \sigma}{\Gamma \rhd \alpha.\rho+\rho' \backsim \overline{\alpha}.\sigma+\sigma'}\,(+\cdot+)$$

If $\models^{\mathbb{AC}}_k \Gamma$ then $\models^{\mathbb{AC}}_{k-1} \Gamma$; by induction over k we know that $\Gamma \models^{\mathbb{AC}}_{k-1} \alpha.\rho+\rho' \backsim \overline{\alpha}.\sigma+\sigma'$ that implies that $\alpha.\rho+\rho'\ \mathbb{AC}_{k-1}\ \overline{\alpha}.\sigma+\sigma'$ and hence $\models^{\mathbb{AC}}_{k-1} \Gamma, \alpha.\rho+\rho' \backsim \overline{\alpha}.\sigma+\sigma'$. From this, by induction over $\mathcal{D}$, we get $\rho\ \mathbb{AC}_{k-1}\ \sigma$; by definition of $\mathbb{AC}_k$ this implies $\alpha.\rho+\rho' \models^{\mathbb{AC}}_k \overline{\alpha}.\sigma+\sigma'$ and we conclude that $\Gamma \models^{\mathbb{AC}}_k \alpha.\rho+\rho' \backsim \overline{\alpha}.\sigma+\sigma'$ as desired.

Cases $(+\cdot\oplus)$ **and** $(+\cdot\oplus)$**:** Similar to case $(+\cdot+)$. □

To get decidability and to prove the completeness property of system $\rhd$, we study the proof-search procedure **Prove** defined in Figure 3. The procedure **Prove**, given a judgment $\Gamma \rhd \rho \backsim \sigma$, attempts to reconstruct a derivation of it in system $\rhd$. Such a procedure is correct and terminating: it either returns a derivation, if any, or it fails, in case the judgment is not derivable in the system.

Lemma 5.7. *The proof search algorithm* **Prove** *for* $\rhd$ *is correct and terminating. In particular,*

(1) $\mathbf{Prove}(\Gamma \rhd \rho \backsim \sigma) = \mathcal{D} \neq \mathbf{fail}$ *implies* $\mathcal{D} :: \Gamma \rhd \rho \backsim \sigma$*;*
(2) $\mathbf{Prove}(\Gamma \rhd \rho \backsim \sigma) = \mathbf{fail}$ *implies* $\Gamma \not\models^{\mathbb{AC}}_k \rho \backsim \sigma$ *for some* k*;*
(3) $\mathbf{Prove}(\Gamma \rhd \rho \backsim \sigma)$ *terminates for all judgments* $\Gamma \rhd \rho \backsim \sigma$*.*

Proof.

(1) Immediate by construction of **Prove**.
(2) Let $\mathbf{Prove}(\Gamma \rhd \rho \backsim \sigma) = \mathbf{fail}$. Then the procedure terminates: let h be the number of nested calls of **Prove** in this execution. We claim that $\Gamma \not\models^{\mathbb{AC}}_{h+1} \rho \backsim \sigma$ which we prove by induction over h. If $h = 0$ then $\rho \backsim \sigma \notin \Gamma$ and none of the conditions defining $\mathbb{AC}_1$ is satisfied. If $h > 0$ then again $\rho \backsim \sigma \notin \Gamma$ and also $\rho \neq \mathbf{1}$. As $h > 0$ there is at least one recursive call of **Prove** and hence either $\rho = \sum_{i\in I} \alpha_i.\rho_i$, $\sigma = \sum_{j\in J} \overline{\alpha}_j.\sigma_j$ and $I \cap J \neq \emptyset$ or $\rho = \bigoplus_{i\in I} \overline{\alpha}_i.\rho_i$, $\sigma = \sum_{j\in J} \alpha_j.\sigma_j$ and $I \subseteq J$ or $\rho = \sum_{i\in I} \overline{\alpha}_i.\rho_i$, $\sigma = \bigoplus_{j\in J} \alpha_j.\sigma_j$ and $I \supseteq J$. In the first case the hypothesis that $\mathbf{Prove}(\Gamma \rhd \rho \backsim \sigma) = \mathbf{fail}$ implies that for all $i \in I \cap J$, $\mathbf{Prove}(\Gamma \rhd \rho_i \backsim \sigma_i) = \mathbf{fail}$; but the number of recursive calls in any call of $\mathbf{Prove}(\Gamma \rhd \rho_i \backsim \sigma_i)$ will be less than h, hence by induction there exists some $l < h$ such that $\Gamma \not\models^{\mathbb{AC}}_{l+1} \rho_i \backsim \sigma_i$, that implies $\Gamma \not\models^{\mathbb{AC}}_h \rho_i \backsim \sigma_i$ for all $i \in I \cap J$ since $\mathbb{AC}_h \subseteq \mathbb{AC}_{l+1}$. It follows that $\Gamma \not\models^{\mathbb{AC}}_{h+1} \rho \backsim \sigma$ by clause (2) in the definition of $\mathbb{AC}_{h+1}$ that is the only one that applies. The other cases of ρ and σ are treated similarly.
(3) Notice that in all recursive calls $\mathbf{Prove}(\Gamma, \rho \backsim \sigma \rhd \rho_k \backsim \sigma_k)$ inside $\mathbf{Prove}(\Gamma \rhd \rho \backsim \sigma)$ the expressions ρ_k and σ_k are subexpressions of ρ and σ respectively. Since contract expressions generate regular trees, there are only finitely many such subexpressions; therefore the if clause $\rho \backsim \sigma \in \Gamma$ corresponding to axiom (Hyp) cannot fail infinitely many times. This implies that the number of nested calls of procedure **Prove** is always finite. □

Decidability now immemdiately descends as a corollary.

Corollary 5.8. *The relation* $\mathbb{AC}$ *is decidable.*

The previous lemma also enables us to get the completeness property.

Proposition 5.9 (Completeness w.r.t $\models^{\mathbb{AC}}$)**.** *If* $\models^{\mathbb{AC}} \rho \backsim \sigma$*, then* $\rhd \rho \backsim \sigma$*.*

Proof. Now suppose that $\rho \,\mathbb{AC}\, \sigma$; by 5.7.(3) the computation of **Prove**$(\emptyset \rhd \rho \rightharpoonup \sigma)$ terminates; the value cannot be **fail** by 5.7.(2), since $\rho \;\mathbb{AC}_k\, \sigma$ for all k by hypothesis, hence the algorithm **Prove** yields a derivation $\mathcal{D}$ of $\emptyset \rhd \rho \rightharpoonup \sigma$ by 5.7.(1). □

5.2. **Characterizations of $\mathbb{AC}$.** The relation $\rho \,\mathbb{AC}\, \sigma$ can be characterized in terms of the existence of a winning strategy for player C in the game $\mathcal{G}_{\rho \| \sigma}$, a condition that in turn is equivalent to ρ and σ being retractable compliant as well as being orchestrated compliant. This is the content of the next theorem, whose proof establishes a tight correspondence among strategies and orchestrators.

Theorem 5.10 (Main Theorem I)**.**
Let $\rho, \sigma \in \mathsf{ASC}$, The following conditions are equivalent:
(1) $\rho \,\mathbb{AC}\, \sigma$
(2) $\rho \dashv^{\mathsf{rtk}} \sigma$
(3) *There exists a winning strategy for player C in $\mathcal{G}_{\rho \,\|\, \sigma}$.*
(4) *There exists an orchestrator f such that $f : \rho \dashv^{\mathsf{Orch}} \sigma$.*

By the above theorem, soundness and completeness of system $\rhd$, as well as the decidability property, immediately transfers from $\mathbb{AC}$ to both $\dashv^{\mathsf{rtk}}$ and $\dashv^{\mathsf{Orch}}$. This might look a bit weird at a first sight, since the judgements of system $\rhd$ abstract away from both histories and orchestrators, which are essential, respectively, for the definition of rollback (and hence of retractable compliance) and for the definition of orchestrated compliance. However, much as it happens with logic, "proofs", namely derivations, can be interpreted as strategies in games determined by their conclusion; on the other hand the informative contents of a derivation lies in the choice of actions and co-actions involved in the interaction among a client and a server, which is exactly the effect of an orchestrator.

The proof of Theorem 5.10 will be developed in Appendix B by proving the following equivalences:

$$(2) \overset{\mathbf{B.1}}{\Leftrightarrow} (1) \overset{\mathbf{B.3}}{\Leftrightarrow} (3)$$
$$\Updownarrow \mathbf{B.4}$$
$$(4)$$

The proofs of such equivalences roughly follow the following schemas.

(1) $\Leftrightarrow$ (2)**:** The relation $\dashv^{\mathsf{rtk}}$ is completely characterized by the properties defining the relation $\mathbb{AC}$, using in an essential way Lemma 3.7.

(1) $\Leftrightarrow$ (3)**:** Since $\mathbb{AC} = \dashv_{\mathsf{tb}}$ by Theorem 2.9, it is enough to show that

$$\rho \dashv_{\mathsf{tb}} \sigma \textit{ if and only if } \textit{there exists a winning strategy for player } \mathsf{C} \textit{ in } \mathcal{G}_{\rho \,\|\, \sigma}$$

This is proved by using a characterization of $\dashv_{\mathsf{tb}}$ in terms of regular trees without "synchronization-failure" leaves. A tree of this sort can be obtained out of a winning strategy, and vice versa.

(1) $\Leftrightarrow$ (4)**:** We provide a formal system $\rhd^{\mathsf{x}}_{\mathsf{o}}$ that is sound and complete with respect to the $\dashv^{\mathsf{Orch}}$ relation. Then we define a procedure that, given a derivation $\mathcal{D} :: \rhd \rho \rightharpoonup \sigma$, returns a derivation $\mathcal{D}' :: \rhd^{\mathsf{x}}_{\mathsf{o}} \mathsf{f}(\mathcal{D}) : \rho \dashv^{\mathsf{Orch}} \sigma$, where $\mathsf{f}(\cdot)$ is a map from derivation to orchestrators, simultaneously defined together with the first procedure.

5.3. **Getting strategies, derivations and orchestrators out of each other.** Strategies, derivations and orchestrators mentioned in the Main Theorem I 5.10 can effectively be computed out of each other as stated in the following theorem.

Theorem 5.11 (Main Theorem II)**.**

(1) *Given a derivation $\mathcal{D}$ for $\rhd \rho \rightharpoonup \sigma$, an orchestrator $f_{\mathcal{D}}$ can be computed out of $\mathcal{D}$, such that $f_{\mathcal{D}} : \rho \dashv^{\mathsf{Orch}} \sigma$;*
(2) *Given an orchestrator f such that $f : \rho \dashv^{\mathsf{Orch}} \sigma$, a strategy Σ_f which is winning for player C in game $\mathcal{G}_{\rho \parallel \sigma}$ can be effectively obtained out of f;*
(3) *Given a winning strategy Σ for player C in the game $\mathcal{G}_{\rho \parallel \sigma}$, an orchestrator f_Σ such that $f_\Sigma : \rho \dashv^{\mathsf{Orch}} \sigma$ can be effectively obtained out of the strategy;*
(4) *Given an orchestrator f such that $f : \rho \dashv^{\mathsf{Orch}} \sigma$, a derivation $\mathcal{D}_f$ for $\rhd \rho \rightharpoonup \sigma$ can be effectively obtained out of f.*

The proof of Main Theorem II is in Appendix C and the related constructions are provided along the following lines:

(1) C.1: We use the function $\mathsf{f}(\cdot)$ from derivations to orchestrators defined in the proof of (1) $\Leftrightarrow$ (4) of Theorem 5.10.
(2) C.2: A "turn-based" version ($\dashv^{\mathsf{Orch}}_{\mathbf{b}}$) of the relation $\dashv^{\mathsf{Orch}}$ is provided at the beginning of Appendix C. Given an orchestrator f such that $f : \rho \dashv^{\mathsf{Orch}}_{\mathbf{b}} \sigma$ (and hence $f : \rho \dashv^{\mathsf{Orch}} \sigma$) we define a procedure yielding a suitable regular tree which can be decorated in order obtain a winning strategy Σ_f for player C in $\mathcal{G}_{\rho \| \sigma}$.
(3) C.3: We use a construction defined in the proof of (1) $\Leftrightarrow$ (3) of Theorem 5.10. Given a univocal winning strategy Σ for player C in the game $\mathcal{G}_{\rho \| \sigma}$ we obtain a tree T representing all the possible plays of the game where player C follows the stategy Σ. Then an orchestrator f_Σ is obtained out of T, such that $f_\Sigma : \rho \dashv^{\mathsf{Orch}}_{\mathbf{b}} \sigma$ (and hence $f_\Sigma : \rho \dashv^{\mathsf{Orch}} \sigma$).
(4) C.4: We define a procedure **O2D** that, given f, ρ and σ such that $f : \rho \dashv^{\mathsf{Orch}} \sigma$, returns a derivation $\mathcal{D}_f :: \rhd \rho \rightharpoonup \sigma$. The procedure **O2D** is obtained by adaptating the proof search procedure **Prove**. In particular, in **O2D** the search is driven by the orchestrator f. Correctness and termination of **O2D** are proved as for **Prove**.

Example 5.12. Let $\mathcal{D}$ be the derivation in Example 5.2. If we consider the function $\mathsf{f}()$ from derivation to orchestrators mentioned before (and formally defined in Appendix B.4, Definition B.12), we have that:

$$\mathsf{f}(\mathcal{D}) = \langle \mathtt{bag}, \overline{\mathtt{bag}} \rangle^{+} . \langle \overline{\mathtt{price}}, \mathtt{price} \rangle . (\langle \mathtt{card}, \overline{\mathtt{card}} \rangle . \mathtt{1} \vee \langle \mathtt{cash}, \overline{\mathtt{cash}} \rangle . \mathtt{1})^{5}$$

If we dub $f = \mathsf{f}(\mathcal{D})$, we have that the strategy Σ_f, obtained by the construction in the proof of (2), is such that

$$\Sigma_f(\boldsymbol{e}) = \begin{cases} \{\, (1, (\mathsf{C}{:}\mathtt{bag})) \,\} & \text{if } \boldsymbol{e} = \langle\rangle \\ \{\, (6, (\mathsf{C}, \checkmark)) \,\} & \text{if } \boldsymbol{e} = \boldsymbol{s_3} \\ \emptyset & \text{for any other play} \end{cases}$$

where $\boldsymbol{s_3} = (1, (\mathsf{C}{:}\mathtt{bag}))(2, (\mathsf{B}{:}\overline{\mathtt{price}}))(3, (\mathsf{A}{:}\mathtt{price}))(4, (\mathsf{A}{:}\overline{\mathtt{cash}}))(5, (\mathsf{B}{:}\mathtt{cash}))$. Observe that Σ_f corresponds to the strategy $\tilde{\Sigma}$ as defined in Example 2.13. The construction of (3), instead, yields f out of Σ_f, whereas $\mathbf{O2D}(f, \mathsf{Buyer}, \mathsf{Seller}) = \mathcal{D}$.

[5]Actually the application of f to the derivation $\mathcal{D}$ does produce some vacuous rec binders. We omit them here for sake of readability.

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

$\quad$ **if** $x : \rho \curlywedge^{\mathsf{Orch}} \sigma \in \Gamma$ **then** $\{ x \}$

$\quad$ **else if** $\rho = \mathbf{1}$ **then** $\{ \mathbf{1} \}$

$\quad$ **else if** $\rho = \bigoplus_{i \in I} \overline{a}_i.\rho_i$ **and** $\sigma = \sum_{j \in I \cup J} a_j.\sigma_j$ **then**
$\quad\quad$ **let** $\Gamma' = \Gamma,\ x{:}\rho \curlywedge^{\mathsf{Orch}} \sigma$ **in**
$\quad\quad\quad \{ \mathsf{rec}\, x. \bigvee_{i \in I} \langle a_i, \overline{a}_i \rangle.f_i \mid \forall i \in I. f_i \in \mathbf{Synth}(\Gamma', \rho_i, \sigma_i \}$

$\quad$ **else if** $\rho = \sum_{j \in I \cup J} \overline{a}_j.\rho_i$ **and** $\sigma = \bigoplus_{i \in I} a_i.\sigma_i$ **then**
$\quad\quad$ **let** $\Gamma' = \Gamma,\ x{:}\rho \curlywedge^{\mathsf{Orch}} \sigma_i$ **in**
$\quad\quad\quad \{ \mathsf{rec}\, x. \bigvee_{i \in I} \langle a_i, \overline{a}_i \rangle.f_i \mid \forall i \in I. f_i \in \mathbf{Synth}(\Gamma', \rho_i, \sigma_i \}$

$\quad$ **else if** $\rho = \sum_{i \in I} \overline{\alpha}_i.\rho_i$ **and** $\sigma = \sum_{j \in J} \alpha_j.\sigma_j$ (where $\alpha \in \mathcal{N} \cup \overline{\mathcal{N}}$) **and** $|I| \geq 2$ **then**
$\quad\quad$ **let** $\Gamma' = \Gamma,\ x{:}\rho \curlywedge^{\mathsf{Orch}} \sigma$ **in**
$\quad\quad\quad \bigcup_{i \in I} \{ \mathsf{rec}\, x. \langle \alpha_i, \overline{\alpha}_i \rangle^+.f \mid f \in \mathbf{Synth}\,(\Gamma', \rho_i, \sigma_i) \}$

$\quad$ **else** $\emptyset$

Figure 4: The algorithm **Synth**.

5.4. **Orchestrator synthesis.** Working on Proposition 5.7 and Theorem 5.11(1) we obtain a synthesis algorithm **Synth** that is defined in Figure 5.4. The algorithm **Synth** takes a (initially empty) set of assumptions Γ and two affectible contracts ρ and σ, and returns a set O of orchestrators (and hence a set of strategies by the above results), if any, such that for any $f \in O$ we have $f : \rho \dashv^{\mathsf{Orch}} \sigma$; the algorithm returns the empty set in case no such an orchestrator exists. In the algorithm **Synth** we consider orchestrators as explicit terms, that is we do not consider recursion up-to recursion unfolding.

Example 5.13. It is not difficult to check that by computing $\mathbf{Synth}(\emptyset, \mathsf{Buyer}, \mathsf{Seller})$ we get a set just consisting of exactly the orchestrator f of Example 4.6, which we have shown to be such that $f : \mathsf{Buyer} \dashv\!\!\!\dashv \mathsf{Seller}$:

$$\mathbf{Synth}(\emptyset,\mathsf{Buyer},\mathsf{Seller}) = \left\{ \langle \mathtt{bag}, \overline{\mathtt{bag}} \rangle^+ . \langle \overline{\mathtt{price}}, \mathtt{price} \rangle (\langle \mathtt{card}, \overline{\mathtt{card}} \rangle . \mathtt{1} \vee \langle \mathtt{cash}, \overline{\mathtt{cash}} \rangle . \mathtt{1}) \right\}$$

The algorithm **Synth** can be proved to be terminating.

Proposition 5.14 (Termination of **Synth**)**.** $\mathbf{Synth}(\Gamma, \rho, \sigma)$ *terminates for any* Γ, ρ *and* σ*.*

Proof. All session contracts in the recursive calls of **Synth** 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 $x : \rho \curlywedge^{\mathsf{Orch}} \sigma \in \Gamma$ in the first clause of **Synth** is always successfully reached in case the algorithm does not terminate because of the last clause. □

The algorithm **Synth** can be proved to be correct and complete in the following sense: whenever it does not fail, it does return a set of correct orchestrators. Moreover, if an orchestrator exists for given ρ and σ, it is actually "captured" by our synthesis algorithm.

Given an orchestrator f we denote by $\mathit{tree}(f)$ its corresponding (possibly infinite) regular tree.

Proposition 5.15 (Correctness and Completeness of **Synth**)**.**
1) *If* $f \in \mathbf{Synth}(\emptyset, \rho, \sigma) \neq \emptyset$ *then* $f : \rho \dashv^{\mathsf{Orch}} \sigma$.
2) *If* $f : \rho \dashv^{\mathsf{Orch}} \sigma$ *then there exists* $g \in \mathbf{Synth}(\emptyset, \rho, \sigma) \neq \emptyset$ *such that with* $tree(f) = tree(g)$.

Proof. See Appendix D. □

6. Subcontract relation: Definition and Main Results.

The notion of compliance naturally induces a substitutability relation on servers that may be used for implementing contract-based query engines (see [16] for a discussion).

Definition 6.1 (Affectible subcontract relation)**.** Let $\sigma, \sigma' \in \mathsf{ASC}$. We define

$$\sigma \preccurlyeq \sigma' \triangleq \forall \rho \, [\, \rho \, \mathbb{AC} \, \sigma \;\Rightarrow\; \rho \, \mathbb{AC} \, \sigma' \,]$$

Example 6.2. Consider the following new version of Seller, that also accepts cheques as payment for the bag and enables customers (may be those with a fidelity card or those who make shopping on Christmas eve) to win the bag by means of a scratch card.

$$\begin{array}{l} \mathsf{SellerII} = \mathtt{belt}.\overline{\mathtt{price}}.\mathtt{cash} \\ \qquad + \\ \qquad \mathtt{bag}.\ (\overline{\mathtt{price}}.(\mathtt{card} + \mathtt{cash} + \mathtt{cheque}) \\ \qquad\quad + \\ \qquad\quad \overline{\mathtt{scratchcard}}) \end{array}$$

It turns out that Seller $\preccurlyeq$ SellerII, so that in particular Buyer $\mathbb{AC}$ SellerII holds.

As done for several notions of compliance for session contracts, decidability of the subcontract relation could be obtained as an immediate consequence of decidability of $\mathbb{AC}$ if we managed to have a proper notion of dual contract and if the following property could be proved:

$$\sigma \preccurlyeq \sigma' \;\Leftrightarrow\; \overline{\sigma} \, \mathbb{AC} \, \sigma' \tag{6.1}$$

However, as already discussed in Remark 1.4, in the present setting the notion of duality is hardly definable so that we have no chance to get (6.1).

Nonetheless decidability of $\preccurlyeq$ can be obtained in a direct way by means of a formal system axiomatising the subcontract relation and of a proof-search algorithm in the style of **Prove**.

6.1. A sound and complete formal system for $\preccurlyeq$.

Definition 6.3 (The Formal System $\blacktriangleright$ for $\preccurlyeq$)**.** A judgment in the formal system $\blacktriangleright$ is an expression of the form $\Gamma \blacktriangleright \rho \ll \sigma$ where Γ is a finite set of expressions with the form $\delta \ll \gamma$, with $\rho, \sigma, \delta, \gamma \in \mathsf{SC}$. Axioms and inference rules of $\blacktriangleright$ are as in Figure 5, where the following provisos hold:

- in rule $(\oplus \cdot + \text{-} \ll)$ we assume that a term of the form $\overline{a}.\sigma_1$ can be used instead of $\overline{a}.\sigma_1 + \sigma'$;
- in rule $(\oplus \cdot + \text{-} \ll)$ we assume either $\sum_{i \in I} \alpha_i.\sigma_i$ or $\sum_{j \in I \cup J} \alpha_j.\sigma'_j$ (not both) can be of the form $\overline{a}.\sigma$[6].

[6]This conditions are needed in order to let the system to be syntax-directed.

$$(\text{Ax-}\ll) : \frac{}{\Gamma \ \blacktriangleright\ \mathbf{1} \ll \sigma'} \qquad (\text{Hyp-}\ll) : \frac{}{\Gamma, \sigma \ll \sigma' \ \blacktriangleright\ \sigma \ll \sigma'}$$

$$(\oplus\cdot+\text{-}\ll) : \frac{\Gamma, \overline{a}.\sigma_1 \oplus \sigma_2 \ll \overline{a}.\sigma_1' + \sigma_2' \ \blacktriangleright\ \sigma_1 \ll \sigma_1'}{\Gamma \ \blacktriangleright\ \overline{a}.\sigma_1 \oplus \sigma_2 \ll \overline{a}.\sigma_1' + \sigma_2'}$$

$$(+\cdot+\text{-}\ll) : \frac{\forall h \in I.\ \Gamma, \sum_{i\in I}\alpha_i.\sigma_i \ll \sum_{j\in I\cup J}\alpha_j.\sigma_j' \ \blacktriangleright\ \sigma_h \ll \sigma_h'}{\Gamma \ \blacktriangleright\ \sum_{i\in I}\alpha_i.\sigma_i \ll \sum_{j\in I\cup J}\alpha_j.\sigma_j'}$$

$$(\oplus\cdot\oplus\text{-}\ll) : \frac{\forall h \in I.\ \Gamma, \bigoplus_{j\in I\cup J}\overline{a}_j.\sigma_j \ll \bigoplus_{i\in I}\overline{a}_i.\sigma_i' \ \blacktriangleright\ \sigma_h \ll \sigma_h'}{\Gamma \ \blacktriangleright\ \bigoplus_{j\in I\cup J}\overline{a}_j.\sigma_j' \ll \bigoplus_{i\in I}\overline{a}_i.\sigma_i'}$$

Figure 5: The formal system $\blacktriangleright$

In system $\blacktriangleright$, the symbol $\ll$ is used as syntactical counterpart of the relation $\preccurlyeq$.

The ideas behind rules $(+\cdot+\text{-}\preccurlyeq)$ and $(\oplus\cdot\oplus\text{-}\preccurlyeq)$ are fairly intuitive. Let us informally see why rule $(\oplus\cdot+\text{-}\preccurlyeq)$ can relate affectible and unaffectible outputs, unlike what happens for other subcontract relations for session contracts. We first observe that a contract ρ which is compliant with a term of the form $\bigoplus_{h\in H}\overline{a}_h.\sigma_h$ must be such that $\rho = \sum_{h\in H'\supseteq H} a_h.\rho_i$ with $\rho_h \dashv\!\shortmid \sigma_h$ for any $h \in H$. A term σ' different from $\bigoplus_{h\in I}\overline{a}_h.\sigma_i$ and such that $\rho \dashv\!\shortmid \sigma'$ can be either of the form $\bigoplus_{k\in K}\overline{a}_k.\sigma_k$, and this case is dealt with by rule $(\oplus\cdot\oplus\text{-}\ll)$; or of the form $\sum_{k\in K}\overline{a}_k.\sigma_k'$. Notice that in order to have $\sum_{h\in H'} a_h.\rho_i \dashv\!\shortmid \sum_{k\in K}\overline{a}_k.\sigma_i''$ it is enough that there exists $p \in H' \cap K$ such that $\rho_p \dashv\!\shortmid \sigma_p'$. This is precisely what is guaranteed by the premise of rule $(+\cdot+\text{-}\preccurlyeq)$.

We can prove system $\blacktriangleright$ to be sound and complete for the subcontract relation $\preccurlyeq$.

Proof search termination for system $\blacktriangleright$ can be shown in the same way as done for $\rhd$.

Proposition 6.4 (Proof search termination)**.** *For system $\blacktriangleright$, proof search does terminate.*

We can now proceed with the soundness and completeness properties for $\blacktriangleright$ with respect to the relation $\preccurlyeq$.

We begin by defining a non-involutive "quasi-dual" operator on affectible contracts, that we shall use to build counterexamples in the proof of Proposition 6.7 below.

Definition 6.5 (An operator of quasi-duality)**.** The operator $\widehat{\cdot} : \mathsf{ASC} \to \mathsf{ASC}$ is inductively defined as follows.

$$\begin{array}{rcl} \widehat{\mathbf{1}} & = & \mathbf{1} \\ \widehat{\bigoplus_{i\in I}\overline{a}_i.\sigma_i} & = & \sum_{i\in I} a_i.\widehat{\sigma}_i \\ \widehat{\sum_{i\in I} a_i.\sigma_i} & = & \bigoplus_{i\in I}\overline{a}_i.\widehat{\sigma}_i \\ \widehat{\sum_{i\in I}\overline{a}_i.\sigma_i} & = & \sum_{i\in I} a_i.\widehat{\sigma}_i \end{array}$$

It is immediate to check that the operator $\widehat{\cdot}$ is not involutive: $\widehat{\widehat{\overline{a}+\overline{b}}} = \widehat{a+b} = \overline{a}\oplus\overline{b}$. However it is enough for us it to enjoy the following property.

Lemma 6.6. *Let $\sigma \in \mathsf{ASC}$.*

$$\widehat{\sigma}\ \mathbb{AC}\ \sigma$$

Proof. By induction on the structure of σ. The base case is immediate. Let us just consider the most interesting case, the other ones being similar.

Case $\sigma = \sum_{i\in I}\overline{a}_i.\sigma_i$**:** By the induction hypothesis we have that $\forall i \in I.\widehat{\sigma_i}\,\mathbb{AC}\,\sigma_i$ (and hence, a fortiori, $\exists k \in (I\cap I).\widehat{\sigma_i}\,\mathbb{AC}\,\sigma_i$). So, we get $\widehat{\sigma} = \sum_{i\in I}a_i.\widehat{\sigma_i}\,\mathbb{AC}\sum_{i\in I}\overline{a}_i.\sigma_i = \sigma$ by definition of $\mathbb{AC}$ (in particular Definition 1.2(2)). □

Proposition 6.7. *$\sigma \preccurlyeq \sigma'$ if and only if one of the following conditions holds:*

(1) $\sigma = \mathbf{1}$*;*
(2) $\sigma = \bigoplus_{j\in J}\overline{a}_j.\sigma_j$*,* $\sigma' = \sum_{i\in I}a_i.\sigma_i$ *and* $\exists k \in (I\cap J) \neq \emptyset.\ \sigma_k \preccurlyeq \sigma'_k$*;*
(3) $\sigma = \sum_{i\in I}\alpha_i.\sigma_i$*,* $\sigma' = \sum_{j\in I\cup J}\alpha_j.\sigma'_j$ *and* $\forall h \in I.\ \sigma_h \preccurlyeq \sigma'_h$*;*
(4) $\sigma = \bigoplus_{j\in I\cup J}\overline{a}_j.\sigma_j$*,* $\sigma' = \bigoplus_{i\in I}\overline{a}_i.\sigma'_i$ *and* $\forall h \in I.\ \rho_h \preccurlyeq \sigma_h$*.*

Proof. ($\Rightarrow$) Let $\sigma \preccurlyeq \sigma'$. Then the only possibilities are necessarily the following ones:

(1) $\sigma = \mathbf{1}$;
(2) $\sigma = \bigoplus_{j\in J}\overline{a}_j.\sigma_j$ and $\sigma' = \sum_{i\in I}\overline{a}_i.\sigma_i$;
(3) $\sigma = \sum_{i\in I}\alpha_i.\sigma_i$ and $\sigma' = \sum_{j\in I\cup J}\alpha_j.\sigma'_j$;
(4) $\sigma = \bigoplus_{j\in I\cup J}\overline{a}_j.\sigma_j$ and $\sigma' = \bigoplus_{i\in I}\overline{a}_i.\sigma'_i$.

We proceed now by cases, according to the shapes of σ and σ'.

(1) Immediate.
(2) We show $\exists k \in I\cap J.\ \sigma_k \preccurlyeq \sigma'_k$ by contradiction. Let us then assume $\forall k \in (I\cap J) \neq \emptyset.\ \sigma_k \not\preccurlyeq \sigma'_k$ and let $\{\,\rho_{p_k}\,\}_{k\in(I\cap J)}$ be such that, for any $k \in (I\cap J)$, $\rho_{p_k}\,\mathbb{AC}\,\sigma$ and $\neg(\rho_{p_k}\,\mathbb{AC}\,\sigma')$. By this and Lemma 6.6. It is easy to check that $\sum_{k\in(I\cap J)}a_k\rho_{p_k} + \sum_{j\in J\setminus(I\cap J)}a_j.\widehat{\sigma}_j\,\mathbb{AC}\,\sigma$, whereas $\neg(\sum_{k\in(I\cap J)}a_k\rho_{p_k} + \sum_{j\in J\setminus(I\cap J)}a_j.\widehat{\sigma}_j\,\mathbb{AC}\,\sigma')$. That is $\sigma \not\preccurlyeq \sigma'$.

The other cases can be proved in a similar way. □

Let us define now a stratified version of $\preccurlyeq$ inspired by Proposition 6.7.

Definition 6.8.

(1) For $n \in \mathbb{N}$, the relation $\preccurlyeq_n \subseteq \mathsf{ASC}\times\mathsf{ASC}$ is defined as follows:
$\preccurlyeq_0 = \mathsf{ASC}\times\mathsf{ASC}$
For $n > 0$, $\mathbf{1} \preccurlyeq_n \sigma$ for any σ, whereas $\sigma \preccurlyeq_n \sigma'$ holds if one of the following conditions holds:
  (a) $\sigma = \bigoplus_{j\in J}\overline{a}_j.\sigma_j$, $\sigma' = \sum_{i\in I}a_i.\sigma_i$ and $\exists k \in (I\cap J) \neq \emptyset.\ \sigma_k \preccurlyeq_{n-1} \sigma'_k$;
  (b) $\sigma = \sum_{i\in I}\alpha_i.\sigma_i$, $\sigma' = \sum_{j\in I\cup J}\alpha_j.\sigma'_j$ and $\forall h \in I.\ \sigma_h \preccurlyeq_{n-1} \sigma'_h$;
  (c) $\sigma = \bigoplus_{j\in I\cup J}\overline{a}_j.\sigma_j$, $\sigma' = \bigoplus_{i\in I}\overline{a}_i.\sigma'_i$ and $\forall h \in I.\ \rho_h \preccurlyeq_{n-1} \sigma_h$.
(2) We define $\preccurlyeq_{co} \triangleq \bigcap_n \preccurlyeq_n$

Lemma 6.9. $\preccurlyeq = \preccurlyeq_{co}$

Definition 6.10 ($\preccurlyeq$-semantics for system $\blacktriangleright$)**.** Let Γ be a set of statements of the form $\rho \ll \sigma$. We define

(1) $\models^{\preccurlyeq} \Gamma$ if $\forall(\rho' \ll \sigma') \in \Gamma.\ [\,\rho' \preccurlyeq \sigma'\,]$;
(2) $\Gamma \models^{\preccurlyeq} \rho \ll \sigma$ if $\models^{\preccurlyeq} \Gamma \Rightarrow \rho \preccurlyeq \sigma$.

Soundness and completeness of $\blacktriangleright$ can hence be formalized as

$$\blacktriangleright\ \sigma \ll \sigma' \ \Leftrightarrow \models^{\preccurlyeq} \sigma \ll \sigma'$$

As done for $\rhd$, we use a stratified version of Definition 6.10.

Definition 6.11 (Stratified $\preccurlyeq$–semantics for $\blacktriangleright$)**.** Let Γ be a set of statements of the form $\rho \ll \sigma$ and let $k \in \mathbb{N}$. We define
(1) $\models^{\preccurlyeq}_k \Gamma \quad \text{if} \quad \forall(\rho' \ll \sigma') \in \Gamma. \quad [\,\rho' \preccurlyeq_k \sigma'\,]$;
(2) $\Gamma \models^{\preccurlyeq}_k \rho \ll \sigma \quad \text{if} \quad \models^{\preccurlyeq}_k \Gamma \;\Rightarrow\; \rho \preccurlyeq_k \sigma$.

It is possible now to verify the following:

Lemma 6.12.
(1) $\preccurlyeq_{k+1} \subseteq \preccurlyeq_k$;
(2) $\models^{\preccurlyeq}_{k+1} \Gamma \;\Rightarrow\; \models^{\preccurlyeq}_k \Gamma$;
(3) $\forall k.\, [\,\Gamma \models^{\preccurlyeq}_k \sigma \ll \sigma'\,] \;\Rightarrow\; \Gamma \models^{\preccurlyeq} \sigma \ll \sigma'$.

Proposition 6.13 (Soundness of $\blacktriangleright$ w.r.t $\models^{\preccurlyeq}$)**.** *If* $\Gamma \blacktriangleright \rho \ll \sigma$*, then* $\Gamma \models^{\preccurlyeq} \rho \ll \sigma$.

Proof. By Lemma 6.12(3), it is enough to show that

$$\Gamma \blacktriangleright \rho \ll \sigma \quad \text{implies} \quad \Gamma \models^{\preccurlyeq}_k \rho \ll \sigma \text{ for all } k.$$

To do that we proceed by simultaneous induction over the derivation $\mathcal{D} :: \Gamma \blacktriangleright \rho \ll \sigma$ and over k.
Since $\Gamma \models^{\preccurlyeq}_0 \rho \ll \sigma$ trivially holds, we shall keep the case $k = 0$ implicit. We distinguish the possible cases of the last rule in $\mathcal{D}$.
Case $(+\cdot+\text{-}\ll)$**:** Then $\mathcal{D}$ ends by

$$\frac{\forall h \in I.\ \Gamma, \sum_{i\in I}\alpha_i.\sigma_i \ll \sum_{j\in I\cup J}\alpha_j.\sigma'_j \blacktriangleright \sigma_h \ll \sigma'_h}{\Gamma \blacktriangleright \sum_{i\in I}\alpha_i.\sigma_i \ll \sum_{j\in I\cup J}\alpha_j.\sigma'_j}$$

We have to prove that $\Gamma \models^{\preccurlyeq}_k \sum_{i\in I}\alpha_i.\sigma_i \ll \sum_{j\in I\cup J}\alpha_j.\sigma'_j$ for all k.

Let $k > 0$; assume, by the induction hypothesis over k, that $\Gamma \models^{\preccurlyeq}_{k-1} \sum_{i\in I}\alpha_i.\sigma_i \ll \sum_{j\in I\cup J}\alpha_j.\sigma'_j$. If $\models^{\preccurlyeq}_k \Gamma$, then $\models^{\preccurlyeq}_{k-1} \Gamma$, which implies $\sum_{i\in I}\alpha_i.\sigma_i \preccurlyeq_{k-1} \sum_{j\in I\cup J}\alpha_j.\sigma'_j$ and hence $\models^{\preccurlyeq}_{k-1} \Gamma, \sum_{i\in I}\alpha_i.\sigma_i \ll \sum_{j\in I\cup J}\alpha_j.\sigma'_j$.
By the induction hypothesis over $\mathcal{D}$ we can hence get that $\sigma_h \preccurlyeq_{k-1} \sigma'_h$ for all h. Now, by Definition 6.8, we get $\sigma \preccurlyeq_k \sigma'$.
The other cases can be proved similarly. □

Completeness can be shown in the same way as done for $\rhd$ with respect to $\mathbb{AC}$.

Proposition 6.14 (Completeness of $\blacktriangleright$ w.r.t $\models^{\preccurlyeq}$)**.** $\sigma \preccurlyeq \sigma' \quad$ *implies* $\quad \blacktriangleright \sigma \ll \sigma'$

We then get decidability as a corollary.

Corollary 6.15 (Decidability of $\preccurlyeq$)**.** *The relation* $\preccurlyeq$ *is decidable.*

Remark 6.16. By Lemma 6.6 it easily descends the following relation between the systems $\rhd$ and $\blacktriangleright$: $\blacktriangleright \sigma \ll \sigma' \;\Rightarrow \rhd \widehat{\sigma} \dashv \sigma'$.
The opposite does not hold. In fact $\overline{a} + \overline{b} = a + b \dashv \overline{a}$, but it is easy to check that $\overline{a} + \overline{b} \not\preccurlyeq \overline{a}$.

Example 6.17. Let us now formally derive that $\mathsf{Seller} \preccurlyeq \mathsf{SellerII}$:

$$\dfrac{\dfrac{\dfrac{\overline{\Gamma_2 \ \blacktriangleright\ \mathbf{1} \ll \mathbf{1}}}{\Gamma_1 \ \blacktriangleright\ \mathtt{csh} \ll \mathtt{csh}}}{\mathsf{S} \ll \mathsf{SII} \ \blacktriangleright\ \overline{\mathtt{pr}}.\mathtt{csh} \ll \overline{\mathtt{pr}}.\mathtt{csh}} \qquad \dfrac{\dfrac{\overline{\Gamma_4 \ \blacktriangleright\ \mathbf{1} \ll \mathbf{1}} \qquad \overline{\Gamma_4 \ \blacktriangleright\ \mathbf{1} \ll \mathbf{1}}}{\Gamma_3 \ \blacktriangleright\ \mathtt{crd} + \mathtt{csh} \ll \mathtt{crd} + \mathtt{csh} + \mathtt{cheque}}}{\mathsf{S} \ll \mathsf{SII} \ \blacktriangleright\ \overline{\mathtt{pr}}.(\mathtt{crd} + \mathtt{csh}) \ll \overline{\mathtt{pr}}.(\mathtt{crd} + \mathtt{csh} + \mathtt{cheque}) + \overline{\mathtt{scratchcard}}}}{\blacktriangleright\ \mathsf{Seller} \ll \mathsf{SellerII}} \tag{6.2}$$

where

$\Gamma_1 = \mathsf{Seller} \ll \mathsf{SellerII},\ \overline{\mathtt{pr}}.\mathtt{csh} \ll \overline{\mathtt{pr}}.\mathtt{csh}$
$\Gamma_2 = \Gamma_1, \mathtt{csh} \ll \mathtt{csh}$
$\Gamma_3 = \mathsf{Seller} \ll \mathsf{SellerII},\ \overline{\mathtt{pr}}.(\mathtt{crd} + \mathtt{csh}) \ll \overline{\mathtt{pr}}.(\mathtt{crd} + \mathtt{csh} + \mathtt{cheque}) + \overline{\mathtt{scratchcard}}$
$\Gamma_4 = \Gamma_3, \mathtt{crd} + \mathtt{csh} \ll \mathtt{crd} + \mathtt{csh} + \mathtt{cheque}$

6.2. **Derivations as orchestrator functors.** By Definition 6.1 and Theorem 5.10, $\sigma \preccurlyeq \sigma'$ implies that, in case there exists f such that $f : \rho \dashv^{\mathsf{Orch}} \sigma$, there exists also f' such that $f' : \rho \dashv^{\mathsf{Orch}} \sigma'$. In the present subsection we show how the computation of f' out of f can be effectively carried out by any derivation of $\blacktriangleright\ \sigma \ll \sigma'$, which can indeed be interpreted as a functor.

In particular, the following definition shows how to get, out of a derivation $\mathcal{D} :: \blacktriangleright\ \sigma \ll \sigma'$, the functor $\mathbf{F}_{\mathcal{D}}$ mapping an orchestrator f such that $f : \rho \dashv^{\mathsf{Orch}} \sigma$ for a certain ρ, to an orchestrator f' such that $f' : \rho \dashv^{\mathsf{Orch}} \sigma$.

In functors, functor variables will be denoted by F, F', ... In order to better grasp how Definition 6.18 below works, we shall use the following equivalent presentation of rule $(\oplus \cdot + \text{-} \ll)$

$$\dfrac{\Gamma, \bigoplus_{i\in I} \overline{a}_i.\sigma_i \ll \sum_{j\in J} \overline{a}_j.\sigma'_j \ \blacktriangleright\ \sigma_k \ll \sigma'_k \qquad (k \in I \cap J) \qquad \mid J \mid \geq 2}{\Gamma \ \blacktriangleright\ \bigoplus_{i\in I} \overline{a}_i.\sigma_i \ll \sum_{j\in J} \overline{a}_j.\sigma'_j} \ (\oplus \cdot + \text{-} \ll)$$

where $\mid J \mid$ denotes the cardinality of J

Besides, we shall use the following two rules instead of the more compact $(+ \cdot + \text{-} \ll)$

$$\dfrac{(\forall h \in I) \qquad \Gamma, \sum_{i\in I} a_i.\sigma_i \ll \sum_{j\in I\cup J} a_j.\sigma'_j \ \blacktriangleright\ \sigma_h \ll \sigma'_h}{\Gamma \ \blacktriangleright\ \sum_{i\in I} a_i.\sigma_i \ll \sum_{j\in I\cup J} a_j.\sigma'_j} \ (+ \cdot + \text{-} \ll \text{-}1)$$

$$\dfrac{(\forall h \in I) \qquad \Gamma, \sum_{i\in I} \overline{a}_i.\sigma_i \ll \sum_{j\in I\cup J} \overline{a}_j.\sigma'_j \ \blacktriangleright\ \sigma_h \ll \sigma'_h}{\Gamma \ \blacktriangleright\ \sum_{i\in I} \overline{a}_i.\sigma_i \ll \sum_{j\in I\cup J} \overline{a}_j.\sigma'_j} \ (+ \cdot + \text{-} \ll \text{-}2)$$

where the same proviso for rule $(+ \cdot + \text{-} \ll)$ (see Definition 6.3) applies.

We call *extended environment* a set of elements of the form $\mathrm{F} : \sigma \ll \sigma'$ where F is an orchestrator variable and $\sigma, \sigma' \in \mathsf{ASC}$. We use the symbols $\widehat{\Gamma}, \widehat{\Gamma}', \ldots$ to range over extended environments.

Definition 6.18. Given $\mathcal{D} :: \Gamma \ \blacktriangleright\ \sigma \ll \sigma'$, the functor $\mathbf{F}_{\mathcal{D}}$ is defined by

$$\mathbf{F}_{\mathcal{D}} = \mathcal{F}(\mathcal{D}, \emptyset)$$

where $\mathcal{F}$ is binary partial function from derivations and extended environments to orchestrator functors. We define $\mathcal{F}$ by induction on the structure of the first argument. The second

argument is used as an accumulator during the construction of the orchestrator functor. It enables us to build recursive orchestrators by keeping the name of recursion variables to be used in case get to an application of rule (Hyp-$\ll$). So a rec binder is introduced in any step of the construction of an orchestrator. Of course the variable introduced will remain vacuously bind in case rule (Hyp-$\ll$) is not encountered. We assume any orchestrator variable introduced to be fresh. $\mathcal{F}(\mathcal{D},\widehat{\Gamma})$ is inductively defined as follows.

$$\mathcal{D} = \frac{}{\Gamma \ \blacktriangleright\ \mathbf{1} \ll \sigma'}\ (\text{Ax-}\ll)\ :$$

$\mathcal{F}(\mathcal{D},\widehat{\Gamma}) = \boldsymbol{\lambda} f.\ \mathtt{1}$

$$\mathcal{D} = \frac{}{\Gamma, \sigma \ll \sigma' \ \blacktriangleright\ \sigma \ll \sigma'}\ (\text{Hyp-}\ll)\ :$$

$\mathcal{F}(\mathcal{D},\widehat{\Gamma}) = \mathrm{F}$
if $\mathrm{F} : \sigma \ll \sigma' \in \widehat{\Gamma}$

$$\mathcal{D} = \frac{\Gamma, \bigoplus_{i\in I}\overline{a}_i.\sigma_i \ll \sum_{j\in J}\overline{a}_j.\sigma'_j \ \blacktriangleright\ \sigma_k \ll \sigma'_k \quad (k \in I \cap J) \quad \mid J \mid \geq 2 \quad \mathcal{D}'}{\Gamma \ \blacktriangleright\ \bigoplus_{i\in I}\overline{a}_i.\sigma_i \ll \sum_{j\in J}\overline{a}_j.\sigma'_j}\ (\oplus\cdot+\text{-}\ll)\ :$$

$\mathcal{F}(\mathcal{D},\widehat{\Gamma}) = \mathsf{rec}\,\mathrm{F}.\boldsymbol{\lambda} f.$ **if** $(f = \bigvee_{h\in H\supseteq I}\langle \overline{a}_h, a_h\rangle.f_h)$ **then** $\langle \overline{a}_k, a_k\rangle^+.\mathcal{F}(\mathcal{D}',\widehat{\Gamma}')(f_k)$ **else** $\mathtt{1}$,
where $\widehat{\Gamma}' = \widehat{\Gamma}, \mathrm{F} : \bigoplus_{i\in I}\overline{a}_i.\sigma_i \ll \sum_{j\in J}\overline{a}_j.\sigma'_j$.

Note that the interaction between a contract and $\bigoplus_{i\in I}\overline{a}_i.\sigma_i$ (with $|I| \geq 2$), in case they are affectible compliant, is mediated by an orchestrator which is necessarily of the form $\bigvee_{h\in H\supseteq I}\langle \overline{a}_h, a_h\rangle.f_h$ or $\mathtt{1}$ (in case the contract is $\mathbf{1}$).

$$\mathcal{D} = \frac{(\forall h \in I) \quad \Gamma, \sum_{i\in I} a_i.\sigma_i \ll \sum_{j\in I\cup J} a_j.\sigma'_j \ \blacktriangleright\ \sigma_h \ll \sigma'_h \quad \mathcal{D}_h}{\Gamma \ \blacktriangleright\ \sum_{i\in I} a_i.\sigma_i \ll \sum_{j\in I\cup J} a_j.\sigma'_j}\ (+\cdot+\text{-}\ll\text{-1})\ :$$

$\mathcal{F}(\mathcal{D},\widehat{\Gamma}) =$
$\mathsf{rec}\,\mathrm{F}.\boldsymbol{\lambda} f.$ **if** $(f = \bigvee_{h\in H'}\langle a_h, \overline{a}_h\rangle.f_h)$
then $\bigvee_{h\in H'\cap I}\langle a_h, \overline{a}_h\rangle.\mathcal{F}(\mathcal{D}_h,\widehat{\Gamma}')(f_h)$
else case f
of $\langle a_k, \overline{a}_k\rangle^+.f'_k \ :\ \langle a_h, \overline{a}_h\rangle^+.\mathcal{F}(\mathcal{D}_h,\widehat{\Gamma}')(f'_h) \qquad (\forall h \in I)$
otherwise $:\ \mathtt{1}$
where $\widehat{\Gamma}' = \widehat{\Gamma}, \mathrm{F} : \sum_{i\in I} a_i.\sigma_i \ll \sum_{j\in I\cup J} a_j.\sigma'_j$

Note that a contract which is in $\mathbb{AC}$ relation with $\sum_{i\in I} a_i.\sigma_i$ is either of the form $\oplus_{h\in H\subseteq I}\overline{a}_h.\sigma'_h$ or of the form $\sum_{h\in H}\overline{a}_h.\sigma'_h$ with $H \cap I \neq \emptyset$. Moreover, in the first case the orchestrator which mediates their interaction has necessarily the form $\bigvee_{h\in H'\supseteq H}\langle a_h, \overline{a}_h\rangle.f_h$; in the second case it has necessarily the form $\langle a_k, \overline{a}_k\rangle^+.f'$ with $k \in I$. We have also to take into account the possibility of f to be $\mathtt{1}$ (in case the contract is $\mathbf{1}$).

$$\mathcal{D} = \dfrac{(\forall h \in I) \quad \overset{\mathcal{D}_h}{\Gamma, \sum_{i\in I}\overline{a}_i.\sigma_i \ll \sum_{j\in I\cup J}\overline{a}_j.\sigma'_j \ \blacktriangleright\ \sigma_h \ll \sigma'_h}}{\Gamma \ \blacktriangleright\ \sum_{i\in I}\overline{a}_i.\sigma_i \ll \sum_{j\in I\cup J}\overline{a}_j.\sigma'_j}\ (+\cdot+\text{-}\ll\text{-}2) : $$

$\mathcal{F}(\mathcal{D}, \widehat{\Gamma}) =$
$\mathsf{rec}\ \mathrm{F}.\boldsymbol{\lambda} f.$ **if** $|I| = 1$ **and** $f = \langle \overline{a}, a\rangle.f' \vee f''$
then $\langle \overline{a}, a\rangle^+.\mathcal{F}(\mathcal{D}', \widehat{\Gamma})(f')$
else case f
of $\langle \overline{a}_h, a_h\rangle^+.f'_h \quad : \quad \langle \overline{a}_h, a_h\rangle^+.\mathcal{F}(\mathcal{D}_h, \widehat{\Gamma}')(f_h) \qquad (\forall h \in I)$
otherwise $\quad : \quad \mathtt{1}$

where $\widehat{\Gamma}' = \widehat{\Gamma}, \mathrm{F} : \sum_{i\in I}\overline{a}_i.\sigma_i \ll \sum_{j\in I\cup J}\overline{a}_j.\sigma'_j$

Note that a contract which is in $\mathbb{AC}$ relation with $\sum_{i\in I}\overline{a}_i.\sigma_i$ is of the form $\sum_{h\in H}a_h.\sigma'_h$ with $I \cap H \neq \emptyset$. Moreover, in case $|I| \geq 2$, the orchestrator which mediates their interaction has necessarily the form $\langle \overline{a}_k, a_k\rangle^+.f'$ with $k \in I \cap H$. If, instead, $|I| = 1$, the orchestrator could also be of the form $\langle \overline{a}, a\rangle.f' \vee f''$. In such a case, the functor has to transform the orchestration action $\langle \overline{a}, a\rangle$ into $\langle \overline{a}, a\rangle^+$ (see Example 6.21 for an example of a case like that). Notice that the present rule cannot be used in case $|I| = |I \cup J| = 1$. We have also to take into account the possibility of f to be $\mathtt{1}$ (in case the contract is $\mathbf{1}$).

$$\mathcal{D} = \dfrac{(\forall h \in I) \quad \overset{\mathcal{D}_h}{\Gamma, \bigoplus_{j\in I\cup J}\overline{a}_j.\sigma_j \ll \bigoplus_{i\in I}\overline{a}_i.\sigma'_i \ \blacktriangleright\ \sigma_h \ll \sigma'_h}}{\Gamma \ \blacktriangleright\ \bigoplus_{j\in I\cup J}\overline{a}_j.\sigma_j \ll \bigoplus_{i\in I}\overline{a}_i.\sigma'_i}\ (\oplus\cdot\oplus\text{-}\ll) : $$

$\mathcal{F}(\mathcal{D}, \widehat{\Gamma}) =$
$\mathsf{rec}\ \mathrm{F}.\boldsymbol{\lambda} f.$ **if** $f = \bigvee_{h\in H'}\langle \overline{a}_h, a_h\rangle.f_h$ **then** $\bigvee_{h\in I}\langle \overline{a}_h, a_h\rangle.\mathcal{F}(\mathcal{D}_h, \widehat{\Gamma}')(f_h)$ **else** $\mathtt{1}$

where $\widehat{\Gamma}' = \widehat{\Gamma}, \mathrm{F} : \bigoplus_{j\in I\cup J}\overline{a}_j.\sigma_j \ll \bigoplus_{i\in I}\overline{a}_i.\sigma'_i$

Note that a contract which is in $\mathbb{AC}$ relation with $\bigoplus_{j\in I\cup J}\overline{a}_j.\sigma_j$ is necessarily of the form $\sum_{h\in H\supseteq I\cup J}a_h.\sigma'_h$. Moreover, the orchestrator which mediates their interaction has necessarily the form $\bigvee_{h\in H'\supseteq I\cup J}\langle \overline{a}_h, a_h\rangle.f_h$. We have also to take into account the possibility of f to be $\mathtt{1}$ (in case the contract is $\mathbf{1}$).

It is easy to check the following.

Fact 6.19. The function $\mathcal{F}$ restricted to empty contexts is total, i.e. given a derivation $\mathcal{D}$ in $\blacktriangleright$, the computation of $\mathbf{F}_{\mathcal{D}}$ always returns an orchestrator functor.

Example 6.20. Let us consider the affectible contracts $\mathtt{d} + \mathtt{b}.(\overline{\mathtt{b}} \oplus \overline{\mathtt{c}})$ and $\mathtt{d}.\overline{\mathtt{a}} + \mathtt{b}.(\overline{\mathtt{a}} + \overline{\mathtt{c}} + \overline{\mathtt{e}})$. Is possible to prove that

$$\mathtt{d} + \mathtt{b}.(\overline{\mathtt{b}} \oplus \overline{\mathtt{c}}) \preccurlyeq \mathtt{d}.\overline{\mathtt{a}} + \mathtt{b}.(\overline{\mathtt{a}} + \overline{\mathtt{c}} + \overline{\mathtt{e}})$$

by means of the following derivation $\mathcal{D}$ in system $\blacktriangleright$.

$$\dfrac{\dfrac{}{\Gamma_1 \blacktriangleright \mathtt{1} \ll \overline{\mathtt{a}}}\,(\text{Ax-}\ll) \qquad \dfrac{\dfrac{\dfrac{}{\Gamma_3 \blacktriangleright \mathtt{1} \ll \mathtt{1}}\,(\text{Ax-}\ll)}{\Gamma_2 \blacktriangleright \overline{\mathtt{c}} \ll \overline{\mathtt{c}}}\,(\oplus\cdot\oplus\text{-}\ll)}{\Gamma_1 \blacktriangleright \overline{\mathtt{b}} \oplus \overline{\mathtt{c}} \ll \overline{\mathtt{a}} + \overline{\mathtt{c}}}\,(\oplus\cdot+\text{-}\ll)}{\blacktriangleright \mathtt{d} + \mathtt{b}.(\overline{\mathtt{b}} \oplus \overline{\mathtt{c}}) \ll \mathtt{d}.\overline{\mathtt{a}} + \mathtt{b}.(\overline{\mathtt{a}} + \overline{\mathtt{c}} + \overline{\mathtt{e}})}\,(+\cdot+\text{-}\ll\text{-1})$$

where

$$\begin{aligned}\Gamma_1 &= \{\, \mathtt{d} + \mathtt{b}.(\overline{\mathtt{b}} \oplus \overline{\mathtt{c}}) \ll \mathtt{d}.\overline{\mathtt{a}} + \mathtt{b}.(\overline{\mathtt{a}} + \overline{\mathtt{c}} + \overline{\mathtt{e}}) \,\}\\ \Gamma_2 &= \Gamma_1, \overline{\mathtt{b}} \oplus \overline{\mathtt{c}} \ll \overline{\mathtt{a}} + \overline{\mathtt{c}}\\ \Gamma_3 &= \Gamma_2, \overline{\mathtt{c}} \ll \overline{\mathtt{c}}\end{aligned}$$

According to Definition 6.18, out of $\mathcal{D}$ we can compute the following functor

$$\begin{aligned}\mathbf{F}_{\mathcal{D}} = &\boldsymbol{\lambda} f.\ \textbf{if}\ (f = \langle \mathtt{d}, \overline{\mathtt{d}}\rangle.f_{\mathtt{d}} \vee \langle \mathtt{b}, \overline{\mathtt{b}}\rangle.f_{\mathtt{b}} \vee f')\\ &\quad \textbf{then}\ \langle \mathtt{d}, \overline{\mathtt{d}}\rangle.\mathtt{1} \vee \langle \mathtt{b}, \overline{\mathtt{b}}\rangle.\ldots.\\ &\quad \textbf{else case}\ f\\ &\qquad \textbf{of}\ \langle \mathtt{d}, \overline{\mathtt{d}}\rangle^{+}.f_{\mathtt{d}} : \langle \mathtt{d}, \overline{\mathtt{d}}\rangle^{+}.\mathtt{1}\\ &\qquad\quad \langle \mathtt{b}, \overline{\mathtt{b}}\rangle^{+}.f_{\mathtt{b}} : \langle \mathtt{b}, \overline{\mathtt{b}}\rangle^{+}.\textbf{if}\ (f_{\mathtt{b}} = \langle \overline{\mathtt{b}}, \mathtt{b}\rangle.f'_{\mathtt{b}} \vee \langle \overline{\mathtt{c}}, \mathtt{c}\rangle.f'_{\mathtt{c}} \vee f'')\\ &\qquad\qquad\qquad\qquad \textbf{then}\ \langle \overline{\mathtt{c}}, \mathtt{c}\rangle^{+}.\mathtt{1}\\ &\qquad\qquad\qquad\qquad \textbf{else}\ \mathtt{1}\\ &\qquad\quad \textbf{otherwise}: \mathtt{1}\end{aligned}$$

(We have omitted the part '...' for sake of readability, as well as the vacuous rec binders.)

It is possible to check that

$$\overline{\mathtt{c}} + \overline{\mathtt{b}}.(\mathtt{b} + \mathtt{c})\ \mathbb{AC}\ \mathtt{d} + \mathtt{b}.(\overline{\mathtt{b}} \oplus \overline{\mathtt{c}})$$

and that

$$\mathsf{f} : \overline{\mathtt{c}} + \overline{\mathtt{b}}.(\mathtt{b} + \mathtt{c}) \dashv^{\mathsf{Orch}} \mathtt{d} + \mathtt{b}.(\overline{\mathtt{b}} \oplus \overline{\mathtt{c}})$$

where $\mathsf{f} = \langle \mathtt{b}, \overline{\mathtt{b}}\rangle^{+}.(\langle \overline{\mathtt{b}}, \mathtt{b}\rangle.\mathtt{1} \vee \langle \overline{\mathtt{c}}, \mathtt{c}\rangle.\mathtt{1})$

We have now that

$$\mathbf{F}_{\mathcal{D}}(\mathsf{f}) = \langle \mathtt{b}, \overline{\mathtt{b}}\rangle^{+}.\langle \overline{\mathtt{c}}, \mathtt{c}\rangle^{+}.\mathtt{1}$$

and $\mathbf{F}_{\mathcal{D}}(\mathsf{f}) : \overline{\mathtt{c}} + \overline{\mathtt{b}}.(\mathtt{b} + \mathtt{c}) \dashv^{\mathsf{Orch}} \mathtt{d}.\overline{\mathtt{a}} + \mathtt{b}.(\overline{\mathtt{a}} + \overline{\mathtt{c}} + \overline{\mathtt{e}}))$

Example 6.21. Let us take $f = \langle \mathtt{bag}, \overline{\mathtt{bag}}\rangle^{+}.\langle \overline{\mathtt{price}}, \mathtt{price}\rangle.(\langle \mathtt{card}, \overline{\mathtt{card}}\rangle.\mathtt{1} \vee \langle \mathtt{cash}, \overline{\mathtt{cash}}\rangle.\mathtt{1})$. We have seen in Example 5.12 that $f : \mathsf{Buyer} \dashv^{\mathsf{Orch}} \mathsf{Seller}$. If we dub now $\mathcal{D}'$ the derivation 6.2 in Example 6.17, we get that

$$\mathbf{F}_{\mathcal{D}'}(f) = \langle \mathtt{bag}, \overline{\mathtt{bag}}\rangle^{+}.\langle \overline{\mathtt{price}}, \mathtt{price}\rangle^{+}.(\langle \mathtt{card}, \overline{\mathtt{card}}\rangle.\mathtt{1} \vee \langle \mathtt{cash}, \overline{\mathtt{cash}}\rangle.\mathtt{1}).$$

Notice how the functor does transform the orchestration action $\langle \overline{\mathtt{price}}, \mathtt{price}\rangle$ of f into the action $\langle \overline{\mathtt{price}}, \mathtt{price}\rangle^{+}$. In fact, whereas $\langle \overline{\mathtt{price}}, \mathtt{price}\rangle$ in f has simply to enable the exchange of the price message, the new orchestrator in that point has to deal with an affectible choice, since also the summand $\overline{\mathtt{scratchcard}}$ is present in SellerII.

It is possible to prove that any functor $\mathbf{F}_{\mathcal{D}}$ behaves as expected.

Theorem 6.22 ($\blacktriangleright$ derivations as orchestrator functors)**.** *Given* $\mathcal{D} :: \blacktriangleright\ \sigma \preccurlyeq \sigma'$*, it is possible to compute a functor* $\mathbf{F}_{\mathcal{D}} : \mathsf{Orch} \to \mathsf{Orch}$ *such that:*

for any ρ *and* f *such that* $f : \rho \dashv^{\mathsf{Orch}} \sigma$*, it holds that* $\mathbf{F}_{\mathcal{D}}(f) : \rho \dashv^{\mathsf{Orch}} \sigma'$*.*

Proof. See Appendix E. □

7. Conclusion and Future Work

We have studied two approaches to loosening compliance among a client and a server in contract theory, based on the concepts of dynamic adaptation and of mediated interaction respectively. We have seen that these induce equivalent notions of compliance, which can be shown via the abstract concept of winning strategy in a suitable class of games.

The byproduct is that the existence of the agreement among two contracts specifying adaptive behaviours is established by statically synthesizing the proper orchestrator, hence avoiding any trial and error mechanism at run time. The study in this paper has been limited to the case of binary sessions since this is the setting in which both orchestrators and retractable contracts have been introduced. However strategy based concepts of agreement have been developed in the more general scenario of multiparty interaction, which seems a natural direction for future work.

Acknowledgments. The authors wish to thank Mariangiola Dezani for her support and Massimo Bartoletti for the preliminary insight that led to the development of the paper.

References

[1] S. Abramsky and P.-A. Mellies. Concurrent games and full completeness. In *Logic in Computer Science, 1999. Proceedings. 14th Symposium on*, pages 431–442, 1999.

[2] Franco Barbanera and Ugo de' Liguoro. Sub-behaviour relations for session-based client/server systems. *MSCS*, 25(6):1339–1381, 2015.

[3] Franco Barbanera and Ugo de' Liguoro. A game interpretation of retractable contracts. In *COORDINATION 2016*, volume 9686 of *Lecture Notes in Computer Science*, pages 18–34. Springer, 2016.

[4] Franco Barbanera, Mariangiola Dezani-Ciancaglini, Ivan Lanese, and Ugo de' Liguoro. Retractable contracts. In *PLACES*, volume 203 of *EPTCS*, pages 61–72. Open Publishing Ass., 2015.

[5] Franco Barbanera, Steffen van Bakel, and Ugo de' Liguoro. Orchestrated session compliance. In Proceedings ICE'15, volume 189 of *EPTCS*, pages 21–36, 2015.

[6] Franco Barbanera, Steffen van Bakel, and Ugo de' Liguoro. Orchestrated session compliance. *Journal of Logical and Algebraic Methods in Programming*, 2016.

[7] Massimo Bartoletti, Tiziana Cimoli, G. Michele Pinna, and Roberto Zunino. Contracts as games on event structures. *J.of Logical and Algebraic Methods in Progr.*, 85(3):399 – 424, 2016.

[8] Giovanni Bernardi and Matthew Hennessy. Compliance and testing preorders differ. In *Software Engineering and Formal Methods - SEFM 2013*, volume 8368 of *LNCS*, pages 69–81, 2013.

[9] Giovanni Tito Bernardi and Matthew Hennessy. Modelling session types using contracts. *Mathematical Structures in Computer Science*, 26(3):510–560, 2016.

[10] Mario Bravetti and Gianluigi Zavattaro. A theory of contracts for strong service compliance. *Mathematical Structures in Computer Science*, 19(3):601–638, 2009.

[11] S. Carpineti, G. Castagna, C. Laneve, and L. Padovani. A formal account of contracts for Web Services. In *WS-FM*, number 4184 in LNCS, pages 148–162. Springer, 2006.

[12] Giuseppe Castagna, Nils Gesbert, and Luca Padovani. A theory of contracts for web services. *ACM Trans. on Prog. Lang. and Sys.*, 31(5):19:1–19:61, 2009.

[13] Simon Gay and Malcolm Hole. Subtyping for Session Types in the Pi-Calculus. *Acta Informatica*, 42(2/3):191–225, 2005.

[14] Cosimo Laneve and Luca Padovani. The Must Preorder Revisited: An Algebraic Theory for Web Services Contracts. In *CONCUR'07*, volume 4703 of *LNCS*, pages 212–225. Springer, 2007.

[15] Luca Padovani. Contract-based discovery and adaptation of web services. In *SFM*, volume 5569 of *LNCS*, pages 213–260. Springer, 2009.

[16] Luca Padovani. Contract-Based Discovery of Web Services Modulo Simple Orchestrators. *Theoretical Computer Science*, 411:3328–3347, 2010.
[17] Glynn Winskel. Event structures. In *Advances in Petri Nets 1986, Part II*, 1987.

Appendix A. Proof of Proposition 2.9 ($\mathbb{AC}$ and $\dashv_{\mathsf{tb}}$ equivalence).

The proof of Theorem 2.9 is developed in the present section along the following lines:

- We first recall the equivalent stratified version of Definition 1.2;
- To each pair $\rho \,\|\, \sigma$ we associate a set of regular trees $\mathsf{rts}(\rho \,\|\, \sigma)$ and show that $\rho \,\mathbb{AC}\, \sigma$ holds if and only if there exists a tree in $\mathsf{rts}(\rho \,\|\, \sigma)$ with no leaf labeled by the symbol $\boldsymbol{\times}$;
- A set $\mathsf{rts}(\rho \,\|\!|\, \sigma)$ of regular trees is also associated to any turn-based system $\rho \,\|\!|\, \sigma$. Also for $\rho \,\|\!|\, \sigma$, we prove $\rho \dashv_{\mathsf{tb}} \sigma$ to hold if and only if there exists a tree in $\mathsf{rts}(\rho \,\|\!|\, \sigma)$ with no leaf labeled by $\boldsymbol{\times}$;
- We conclude by showing how to map $\mathsf{rts}(\rho \,\|\, \sigma)$ to $\mathsf{rts}(\rho \,\|\!|\, \sigma)$ and vice versa so that a tree without $\boldsymbol{\times}$ is always sent to a tree with the same property.

In Definition 5.3 we have seen that the co-inductive definition of $\mathbb{AC}$ (Def. 1.2) can be stratified by the family $\{\, \mathbb{AC}_k \,\}_{k\in\mathbb{N}}$, such that $\mathbb{AC}_k \subseteq \mathbb{AC}_{k-1}$ for all k and $\mathbb{AC} = \bigcap_{n\in\mathbb{N}} \mathbb{AC}_n$.

Definition A.1. Let $\rho, \sigma \in \mathsf{ASC}$. We define the set of regular trees of the pair $\rho \,\|\, \sigma$, which we dub $\mathsf{rts}(\rho \,\|\, \sigma)$, as follows:

$$\mathsf{rts}(\mathbf{1} \,\|\, \sigma) \;=\; \left\{\; \mathbf{1} \,\|\, \sigma \;\right\}$$

$$\mathsf{rts}(\rho \,\|\, \sigma) \;=\; \left\{ \begin{array}{c} \rho \,\|\, \sigma \\ /\cdots\backslash \\ \mathsf{T}_1 \cdots\ \mathsf{T}_n \end{array} \;\middle|\; h \in I, \mathsf{T}_h \in \mathsf{rts}(\rho_h \,\|\, \sigma_h) \right\}$$
$$\text{if either } \rho = \textstyle\bigoplus_{i\in I} \overline{a}_i.\rho_i,\ \sigma = \sum_{j\in J} a_j.\sigma_j,\ I \subseteq J$$
$$\text{or } \quad \rho = \textstyle\sum_{j\in J} a_j.\rho_j,\ \sigma = \bigoplus_{i\in I} \overline{a}_i.\sigma_i,\ I \subseteq J$$
$$\text{where } I = \{\, 1, .., n \,\}.$$

$$\mathsf{rts}(\rho \,\|\, \sigma) \;=\; \textstyle\bigcup_{h\in I\cap J} \left\{ \begin{array}{c} \rho \,\|\, \sigma \\ | \\ \mathsf{T} \end{array} \;\middle|\; \mathsf{T} \in \mathsf{rts}(\rho_h \,\|\, \sigma_h) \right\}$$
$$\text{if } \rho = \textstyle\sum_{i\in I} \alpha_i.\rho_i,\ \sigma = \sum_{j\in J} \overline{\alpha}_j.\sigma_j,\ h \in I \cap J$$

$$\mathsf{rts}(\rho \,\|\, \sigma) \;=\; \left\{ \begin{array}{c} \rho \,\|\, \sigma \\ | \\ \boldsymbol{\times} \end{array} \right\} \text{ otherwise}$$

Lemma A.2. *Let $\rho, \sigma \in \mathsf{ASC}$. $\rho \,\mathbb{AC}\, \sigma$ iff there exists a tree in $\mathsf{rts}(\rho \,\|\, \sigma)$ without any leaf labeled by $\boldsymbol{\times}$.*

Proof. ($\Leftarrow$) Let $\mathsf{T} \in \mathsf{rts}(\rho \,\|\, \sigma)$ such that no leaf is labeled by $\boldsymbol{\times}$ and consider the following relation:

$$\mathcal{R} = \{\, (\rho', \sigma') \mid \rho' \,\|\, \sigma' \text{ is a node of } \mathsf{T} \,\}$$

Then $\rho \mathcal{R} \sigma$ as $\rho \parallel \sigma$ is the label of the root of T. Besides, it is easy to check that $\mathcal{R}$ is an affectible compliance relation according to Definition 5.3.

($\Rightarrow$) Recall that $\mathbb{AC} = \bigcap_{k\in\mathbb{N}} \mathbb{AC}_k$. Then we prove by induction over k that if $\rho\, \mathbb{AC}_k \sigma$ then there exists a tree $\mathsf{T} \in \mathsf{rts}(\rho \parallel \sigma)$ such the cut of T at level k, written $\mathsf{T}{\downarrow_k}$ has no leaf labelled by $\bigstar$.

If $k = 0$ we just observe that $\mathsf{T}{\downarrow_0}$ is the root of a tree $\mathsf{T} \in \mathsf{rts}(\rho \parallel \sigma)$, that cannot be labelled by $\bigstar$.

Let $k > 0$, then we proceed according to the cases in the definition of T. If $\mathsf{T}{\downarrow_k} = \mathbf{1} \parallel \sigma$ there is nothing to prove. The case $\mathsf{T}{\downarrow_k} = \left\{ \begin{array}{c} \rho \parallel \sigma \\ | \\ \bigstar \end{array} \right\}$ is impossible since it implies that none of the cases of Definition 5.3(1) holds. This contradicts $\rho\, \mathbb{AC}_k \sigma$, as $k > 0$.

Suppose that $\mathsf{T} = \begin{array}{c} \rho \parallel \sigma \\ /\cdots\backslash \\ \mathsf{T}_1 \cdots\ \mathsf{T}_n \end{array} \in \mathsf{rts}(\rho \parallel \sigma)$, where ρ and σ are as described in the definition. The hypothesis $\rho\, \mathbb{AC}_k \sigma$ implies that $\rho_h\, \mathbb{AC}_{k-1} \sigma_h$ for all h; hence by induction, for all h there exists $\mathsf{T}'_h \in \mathsf{rts}(\rho_h \parallel \sigma_h)$ such that $\mathsf{T}'_h{\downarrow_{k-1}}$ has no leaf labelled by $\bigstar$. It follows that $\mathsf{T}' = \begin{array}{c} \rho \parallel \sigma \\ /\cdots\backslash \\ \mathsf{T}'_1 \cdots\ \mathsf{T}'_n \end{array}$ is a tree in $\mathsf{rts}(\rho \parallel \sigma)$, such that $\mathsf{T}'{\downarrow_k}$ has no leaf labelled by $\bigstar$.

Finally if $\mathsf{T} = \begin{array}{c} \rho \parallel \sigma \\ | \\ \mathsf{T}_h \end{array}$ where ρ and σ are as in the third case of the definition and $h \in I \cap J$ and $\mathsf{T}_h \in \mathsf{rts}(\rho_h \parallel \sigma_h)$, then by induction and reasoning as in the previous case we get a tree $\mathsf{T}'_h \in \mathsf{rts}(\rho_h \parallel \sigma_h)$ with no leaf labelled by $\bigstar$. Hence $\mathsf{T}' = \begin{array}{c} \rho \parallel \sigma \\ | \\ \mathsf{T}'_h \end{array} \in \mathsf{rts}(\rho \parallel \sigma)$ is such that $\mathsf{T}'{\downarrow_k}$ has no leaf labelled by $\bigstar$. □

Definition A.3. Let $\tilde\rho, \tilde\sigma \in \mathsf{ASC}^{[]}$. We define the set of regular trees of the system $\tilde\rho \lll \tilde\sigma$, which we dub $\mathsf{rts}(\tilde\rho \lll \tilde\sigma)$; let $B = \{\mathsf{A}{:}a, \mathsf{A}{:}\overline{a}, \mathsf{B}{:}a, \mathsf{B}{:}\overline{a} \mid a \in \mathcal{N}\}$, then:

$$\mathsf{rts}(\mathbf{1} \lll \tilde\sigma) = \left\{ \begin{array}{c} \mathbf{1} \lll \tilde\sigma \\ | \\ \mathbf{0} \lll \tilde\sigma \end{array} \right\}$$

$$\mathsf{rts}(\tilde\rho \lll \tilde\sigma) = \left\{ \begin{array}{c} \tilde\rho \lll \tilde\sigma \\ /\cdots\backslash \\ \mathsf{T}_1 \cdots\ \mathsf{T}_n \end{array} \;\middle|\; \mathsf{T}_1 \in \mathsf{rts}(\tilde\rho_1 \lll \tilde\sigma_1), \ldots, \mathsf{T}_n \in \mathsf{rts}(\tilde\rho_n \lll \tilde\sigma_n) \right\}$$
$$\text{where } \{\, \tilde\rho_i \lll \tilde\sigma_i \,\}_{i=1..n} = \{\, \tilde\rho' \lll \tilde\sigma' \;\mid\; \tilde\rho \lll \tilde\sigma \xrightarrow{\beta}\!\!\!\!\rightarrow\; \tilde\rho' \lll \tilde\sigma',\ \beta \in B \,\} \neq \emptyset$$

$$\mathsf{rts}(\tilde\rho \lll \tilde\sigma) = \left\{ \begin{array}{c} \tilde\rho \lll \tilde\sigma \\ | \\ \mathsf{T} \end{array} \;\middle|\; \exists \tilde\rho', \tilde\sigma', a.\ \tilde\rho \lll \tilde\sigma \xrightarrow{\mathsf{C}:a}\!\!\!\!\rightarrow\; \tilde\rho' \lll \tilde\sigma', \mathsf{T} \in \mathsf{rts}(\tilde\rho' \lll \tilde\sigma') \right\}$$

$$\mathsf{rts}(\tilde\rho \lll \tilde\sigma) = \left\{ \begin{array}{c} \tilde\rho \lll \tilde\sigma \\ | \\ \bigstar \end{array} \right\} \text{ if } \rho \neq \mathbf{1} \text{ and } \tilde\rho \lll \tilde\sigma \not\longrightarrow\!\!\!\!\rightarrow$$

Lemma A.4. *Let $\rho, \sigma \in \mathsf{ASC} (\subseteq \mathsf{ASC}^{[]})$.*
$\rho \dashv_{\mathsf{tb}} \sigma$ iff there exists a tree in $\mathsf{rts}(\rho \parallel \sigma)$ such that no leaf is labeled by $\boldsymbol{\times}$.

Proof. Similar to the proof of Lemma A.2. □

Next we define a function $\mathsf{to}_{\parallel} \colon \mathsf{rts}(\rho \parallel \sigma) \to \mathsf{rts}(\rho \,\|\, \sigma)$, for which some technical results are in order.

Lemma A.5. *Let $\rho, \sigma \in \mathsf{ASC} \subseteq \mathsf{ASC}^{[]}$ and let $\tilde{\rho}, \tilde{\sigma} \in \mathsf{ASC}^{[]}$ such that $\rho \parallel \sigma \longrightarrow\!\!\!\!\!\rightarrow^{*} \tilde{\rho} \parallel \tilde{\sigma}$.*

(1) *If $\tilde{\rho} \parallel \tilde{\sigma} \xrightarrow{\beta}\!\!\!\!\!\rightarrow$ then only one of the following cases can occur:*
   (a) *$\beta \in \{\mathsf{A} : a, \mathsf{B} : a \mid a \in \mathcal{N}\}$ and β is unique;*
   (b) *$\beta \in \{\mathsf{A} : \overline{a}, \mathsf{B} : \overline{a} \mid a \in \mathcal{N}\}$;*
   (c) *$\beta \in \{\mathsf{C} : a \mid a \in \mathcal{N}\}$ and β is unique.*

(2) *If $\tilde{\rho} \dashv_{\mathsf{tb}} \tilde{\sigma}$ and $\tilde{\rho} \parallel \tilde{\sigma} \xrightarrow{\beta}\!\!\!\!\!\rightarrow \tilde{\rho}' \parallel \tilde{\sigma}' \longrightarrow\!\!\!\!\!\rightarrow$, where $\beta \in \{\mathsf{A} : \overline{a}, \mathsf{B} : \overline{a} \mid a \in \mathcal{N}\}$, then $\tilde{\rho}, \tilde{\sigma} \in \mathsf{ASC}$ and there exists a unique $\beta' \in \{\mathsf{A} : a, \mathsf{B} : a \mid a \in \mathcal{N}\}$ such that $\tilde{\rho}' \parallel \tilde{\sigma}' \xrightarrow{\beta'}\!\!\!\!\!\rightarrow \rho' \parallel \sigma'$ with $\rho', \sigma' \in \mathsf{ASC}$. Moreover, $\tilde{\rho} \parallel \tilde{\sigma} \Longrightarrow \rho' \parallel \sigma'$*

Proof. Immediate by definition of $\mathsf{ASC}^{[]}$ and $\longrightarrow\!\!\!\!\!\rightarrow$. □

Definition A.6. Let $\mathsf{T} \in \mathsf{rts}(\rho \parallel \sigma)$ where $\rho, \sigma \in \mathsf{ASC}$. We define the regular tree $\mathsf{to}_{\parallel}(\mathsf{T})$ as follows:
let $B = \{\mathsf{A}{:}a, \mathsf{B}{:}a \mid a \in \mathcal{N}\}$ and $\overline{B} = \{\mathsf{A}{:}\overline{a}, \mathsf{B}{:}\overline{a} \mid a \in \mathcal{N}\}$

$$\mathsf{to}_{\parallel}\Big(\begin{array}{c} \mathbf{1} \parallel \tilde{\sigma} \\ | \\ \mathbf{0} \parallel \tilde{\sigma} \end{array}\Big) = \mathbf{1} \,\|\, \tilde{\sigma}$$

$$\mathsf{to}_{\parallel}\Big(\begin{array}{c} \rho \parallel \sigma \\ /\cdots\backslash \\ [\overline{a}_1].\rho_1 \parallel \sigma \cdots [\overline{a}_n].\rho_n \parallel \sigma \\ | \quad \cdots \quad | \\ \mathsf{T}_1 \cdots \mathsf{T}_n \end{array}\Big) = \begin{array}{c} \rho \,\|\, \sigma \\ /\cdots\backslash \\ \mathsf{to}_{\parallel}(\mathsf{T}_1) \cdots \mathsf{to}_{\parallel}(\mathsf{T}_n) \end{array}$$

if $\rho \parallel \sigma \xrightarrow{\mathsf{A}:\overline{a}_k}\!\!\!\!\!\rightarrow [\overline{a}_k]\rho_k \parallel \tilde{\sigma}$ for $k = 1, \ldots, n$,
and similarly if $\rho \parallel \sigma \xrightarrow{\mathsf{A}:\overline{a}_k}\!\!\!\!\!\rightarrow \tilde{\rho} \parallel [\overline{a}_k]\sigma_k$.

$$\mathsf{to}_{\parallel}\Big(\begin{array}{c} \rho \parallel \sigma \\ | \\ \mathsf{T}' \end{array}\Big) = \begin{array}{c} \rho \,\|\, \sigma \\ | \\ \mathsf{to}_{\parallel}(\mathsf{T}') \end{array} \quad \text{if} \quad \begin{array}{c} \exists a \in \mathcal{N}.\, \rho \parallel \sigma \xrightarrow{\mathsf{C}:a}\!\!\!\!\!\rightarrow \\ \text{or} \\ \exists \beta \in B.\, \rho \parallel \sigma \xrightarrow{\beta}\!\!\!\!\!\rightarrow \end{array}$$

$$\mathsf{to}_{\parallel}\Big(\begin{array}{c} \tilde{\rho} \parallel \tilde{\sigma} \\ | \\ \boldsymbol{\times} \end{array}\Big) = \begin{array}{c} \tilde{\rho} \,\|\, \tilde{\sigma} \\ | \\ \boldsymbol{\times} \end{array} \quad \text{else}$$

Lemma A.7. *Let $\mathsf{T} \in \mathsf{rts}(\rho \parallel \sigma)$ where $\rho, \sigma \in \mathsf{ASC}$ and such that all its leaves are of the form $\mathbf{0} \parallel \tilde{\sigma}$. Then all leaves of $\mathsf{to}_{\parallel}(\mathsf{T})$ are of the form $\mathbf{1} \,\|\, \tilde{\sigma}$.*

Proof. By definition of $\mathsf{to}_{\parallel}$. □

We define now a function $\mathsf{to}_{\|\!\|\!\|} \colon \mathsf{rts}(\rho \,\|\, \sigma) \to \mathsf{rts}(\rho \,|\!|\!|\, \sigma)$. The following facts are immediate by definition.

Lemma A.8. *Let $\rho, \sigma, \rho_h, \sigma_h \in \mathsf{ASC}$ have the form as in (1b) or (1c) of 5.3, where $h \in I$. Then $\rho \,\|\, \sigma \xrightarrow{X:\overline{a}} \tilde{\rho} \,\|\, \tilde{\sigma} \xrightarrow{\overline{X}:\overline{a}} \rho_h \,\|\, \sigma_h$ for some $\tilde{\rho}, \tilde{\sigma} \in \mathsf{ASC}^{[]}$, $a \in \mathcal{N}$ and $X \in \{\mathsf{A}, \mathsf{B}\}$, where $\overline{X} = \mathsf{A}$ if $X = \mathsf{B}$ and $\overline{X} = \mathsf{B}$ if $X = \mathsf{A}$.*

Lemma A.9. *Let $\rho, \sigma, \rho_h, \sigma_h \in \mathsf{ASC}$ have the form as in (1a) of 5.3, where $h \in I$. Then $\rho \,\|\, \sigma \xrightarrow{\mathsf{C}:a} \rho_h \,\|\, \sigma_h$ for some $a \in \mathcal{N}$.*

Definition A.10. Let $\mathsf{T} \in \mathsf{rts}(\rho \,\|\, \sigma)$ where $\rho, \sigma \in \mathsf{ASC}$. We define the regular tree $\mathsf{to}_{|\!|\!|}(\mathsf{T})$ as follows:

$$\mathsf{to}_{|\!|\!|}(\mathbf{1} \,\|\, \tilde{\sigma}) = \begin{array}{c} \mathbf{1} \,|\!|\!|\, \tilde{\sigma} \\ | \\ \mathbf{0} \,|\!|\!|\, \tilde{\sigma} \end{array}$$

$$\mathsf{to}_{|\!|\!|}\Big(\begin{array}{c} \rho \,\|\, \sigma \\ /\cdots\backslash \\ \mathsf{T}_1 \cdots\ \mathsf{T}_n \end{array}\Big) = \begin{array}{c} \rho \,|\!|\!|\, \sigma \\ /\cdots\backslash \\ \tilde{\rho}_1 \,|\!|\!|\, \tilde{\sigma}_1 \cdots \tilde{\rho}_n \,|\!|\!|\, \tilde{\sigma}_n \\ |\ \cdots\ | \\ \mathsf{to}_{|\!|\!|}(\mathsf{T}_1) \cdots \mathsf{to}_{|\!|\!|}(\mathsf{T}_n) \end{array} \quad \text{if } \rho \text{ and } \sigma \text{ are as in (1b) or (1c) of 5.3}$$

where $\{\, \tilde{\rho}_i \,|\!|\!|\, \tilde{\sigma}_i \,\}_{i=1..n} = \{\, \tilde{\rho}' \,|\!|\!|\, \tilde{\sigma}' \;\mid\; \rho \,|\!|\!|\, \sigma \xrightarrow{\beta} \tilde{\rho}' \,|\!|\!|\, \tilde{\sigma}', \beta \in \overline{B} \,\}$
with $\overline{B} = \{\mathsf{A}{:}\overline{a}, \mathsf{B}{:}\overline{a} \mid a \in \mathcal{N}\}$

$$\mathsf{to}_{|\!|\!|}\Big(\begin{array}{c} \rho \,\|\, \sigma \\ | \\ \mathsf{T}' \end{array}\Big) = \begin{array}{c} \rho \,|\!|\!|\, \sigma \\ | \\ \mathsf{to}_{|\!|\!|}(\mathsf{T}') \end{array} \quad \text{if } \rho \text{ and } \sigma \text{ are as in (1a) of 5.3}$$

$$\mathsf{to}_{|\!|\!|}\Big(\begin{array}{c} \tilde{\rho} \,\|\, \tilde{\sigma} \\ | \\ \boldsymbol{\times} \end{array}\Big) = \begin{array}{c} \tilde{\rho} \,|\!|\!|\, \tilde{\sigma} \\ | \\ \boldsymbol{\times} \end{array}$$

Lemma A.11. *Let $\mathsf{T} \in \mathsf{rts}(\rho \,\|\, \sigma)$ where $\rho, \sigma \in \mathsf{ASC}$ and such that all its leaves are of the form $\mathbf{1} \,\|\, \tilde{\sigma}$. Then all the leaves of $\mathsf{to}_{|\!|\!|}(\mathsf{T})$ are of the form $\mathbf{0} \,|\!|\!|\, \tilde{\sigma}$.*

Proof. By definition of $\mathsf{to}_{|\!|\!|}$. □

The following immediately descends from Lemmas A.7 and A.11.

Theorem A.12. *$\rho, \sigma \in \mathsf{ASC}(\subset \mathsf{ASC}^{[]})$.*

(1) *Let $\mathsf{T} \in \mathsf{rts}(\rho \,|\!|\!|\, \sigma)$ such that all its leaves are of the form $\mathbf{0} \,|\!|\!|\, \tilde{\sigma}$, then there exists $\mathsf{T}' \in \mathsf{rts}(\rho \,\|\, \sigma)$ such that all its leaves are of the form $\mathbf{1} \,\|\, \tilde{\sigma}$.*
(2) *Let $\mathsf{T} \in \mathsf{rts}(\rho \,\|\, \sigma)$ such that all its leaves are of the form $\mathbf{1} \,\|\, \tilde{\sigma}$, then there exists $\mathsf{T}' \in \mathsf{rts}(\rho \,|\!|\!|\, \sigma)$ such that all its leaves are of the form $\mathbf{0} \,|\!|\!|\, \tilde{\sigma}$.*

We then get Theorem 2.9 as a corollary of Lemmas A.4, A.2 and Theorem A.12.

Appendix B. Proof of Main Theorem I (Theorem 5.10)

B.1. **Proof of** (1) $\Leftrightarrow$ (2) **of Theorem 5.10 (** $\dashv^{\mathsf{rdk}} = \mathbb{AC}$**).**

In order to prove the equivalence of items (1) and (2) of the Main Theorem I (Theorem 5.10), that is

$$\dashv^{\mathsf{rdk}} = \mathbb{AC},$$

we need to prove that $\dashv^{\mathsf{rdk}}$ satifies the properties in Lemma 3.7.

B.2. **Proof of Lemma 3.7 (**[4]**) (Rollback properties).** [7]

Lemma 3.7(1) If $[\,] \prec \rho \parallel [\,] \prec \sigma \xrightarrow{*} \boldsymbol{\delta} \prec \rho' \parallel \boldsymbol{\gamma} \prec \sigma' \not\rightarrow$, then $\boldsymbol{\delta} = \boldsymbol{\gamma} = [\,]$.

Proof. Clearly $\boldsymbol{\delta} \prec \rho' \parallel \boldsymbol{\gamma} \prec \sigma' \not\rightarrow$ implies either $\boldsymbol{\delta} = [\,]$ or $\boldsymbol{\gamma} = [\,]$. Observe that:

- rule (comm) adds one element to both stacks;
- rule (τ) does not modify both stacks;
- rule (rbk) removes one element from both stacks. □

Lemma 3.7(2) If $\boldsymbol{\delta} \prec \rho \dashv^{\mathsf{rdk}} \boldsymbol{\gamma} \prec \sigma$, then $\boldsymbol{\delta'} : \boldsymbol{\delta} \prec \rho \dashv^{\mathsf{rdk}} \boldsymbol{\gamma'} : \boldsymbol{\gamma} \prec \sigma$ for all $\boldsymbol{\delta'}$, $\boldsymbol{\gamma'}$.

Proof. It suffices to show that

$$\boldsymbol{\delta} \prec \rho \dashv^{\mathsf{rdk}} \boldsymbol{\gamma} \prec \sigma \;\Rightarrow\; \rho' : \boldsymbol{\delta} \prec \rho \dashv^{\mathsf{rdk}} \boldsymbol{\gamma} \prec \sigma \text{ and } \boldsymbol{\delta} \prec \rho \dashv^{\mathsf{rdk}} \sigma' : \boldsymbol{\gamma} \prec \sigma$$

which we prove by contraposition.

Suppose that $\rho' : \boldsymbol{\delta} \prec \rho \not\dashv^{\mathsf{rdk}} \boldsymbol{\gamma} \prec \sigma$; then

$$\rho' : \boldsymbol{\delta} \prec \rho \parallel \boldsymbol{\gamma} \prec \sigma \xrightarrow{*} \boldsymbol{\delta}' \prec \rho'' \parallel \boldsymbol{\gamma}' \prec \sigma'' \not\rightarrow \text{ and } \rho'' \neq \mathbf{1}$$

If ρ' is never used, then $\boldsymbol{\delta}' = \rho' : \boldsymbol{\delta}''$ and $\boldsymbol{\gamma}' = [\,]$, so that we get

$$\boldsymbol{\delta} \prec \rho \parallel \boldsymbol{\gamma} \prec \sigma \xrightarrow{*} \boldsymbol{\delta}'' \prec \rho'' \parallel [\,] \prec \sigma'' \not\rightarrow$$

Otherwise we have that

$$\rho' : \boldsymbol{\delta} \prec \rho \parallel \boldsymbol{\gamma} \prec \sigma \xrightarrow{*} \rho' \prec \rho'' \parallel \boldsymbol{\gamma}' \prec \sigma'' \longrightarrow [\,] \prec \rho' \parallel \boldsymbol{\gamma}'' \prec \sigma'''$$

and we assume that $\xrightarrow{*}$ is the shortest such reduction. It follows that $\rho'' \neq \mathbf{1}$. By the minimality assumption about the length of $\xrightarrow{*}$ we know that ρ' neither has been restored by some previous application of rule (rbk), nor pushed back into the stack before. We get

$$\boldsymbol{\delta} \prec \rho \parallel \boldsymbol{\gamma} \prec \sigma \xrightarrow{*} [\,] \prec \rho'' \parallel \boldsymbol{\gamma}'' \prec \sigma'' \not\rightarrow$$

In both cases we conclude that $\boldsymbol{\delta} \prec \rho \not\dashv^{\mathsf{rdk}} \boldsymbol{\gamma} \prec \sigma$ as desired.

Similarly we can show that $\boldsymbol{\delta} \prec \rho \not\dashv^{\mathsf{rdk}} \sigma' : \boldsymbol{\gamma} \prec \sigma \;\Rightarrow\; \boldsymbol{\delta} \prec \rho \not\dashv^{\mathsf{rdk}} \boldsymbol{\gamma} \prec \sigma$. □

[7] The proof of Lemma 3.7 is from the workshop paper [4]. We restate it here for the reader's convenience.

We can now show that the rollback compliance and the relation $\mathbb{AC}$ do coincide.

Lemma B.1. *We have $\rho \dashv^{\mathsf{rbk}} \sigma$ if and only if one of the following conditions holds:*
(1) $\rho = \mathbf{1}$;
(2) $\rho = \sum_{i\in I} \alpha_i.\rho_i$, $\sigma = \sum_{j\in J} \overline{\alpha}_j.\sigma_j$ *and* $\exists k \in I \cap J.\ \rho_k \dashv^{\mathsf{rbk}} \sigma_k$;
(3) $\rho = \bigoplus_{i\in I} \overline{\alpha}_i.\rho_i$, $\sigma = \sum_{j\in J} \alpha_j.\sigma_j$, $I \subseteq J$ *and* $\forall k \in I.\ \rho_k \dashv^{\mathsf{rbk}} \sigma_k$;
(4) $\rho = \sum_{i\in I} \overline{\alpha}_i.\rho_i$, $\sigma = \bigoplus_{j\in J} \alpha_j.\sigma_j$, $I \supseteq J$ *and* $\forall k \in J.\ \rho_k \dashv^{\mathsf{rbk}} \sigma_k$.

Proof. ($\Leftarrow$) Immediate.
($\Rightarrow$)[8] We prove this by contraposition and by cases according to the possible shapes of ρ and σ in the conditions of Definition 1.2. Suppose $\rho = \sum_{i\in I} \alpha_i.\rho_i$, $\sigma = \sum_{j\in J} \overline{\alpha}_j.\sigma_j$, $I \cap J = \{k_1, \ldots, k_n\}$ and $\rho_{k_i} \not\dashv^{\mathsf{rbk}} \sigma_{k_i}$ for $1 \leq i \leq n$. Then we get

$$[\,]\prec \rho_{k_i} \parallel [\,]\prec \sigma_{k_i} \xrightarrow{*} \boldsymbol{\delta}_i \prec \rho_i' \parallel \boldsymbol{\gamma}_i \prec \sigma_i' \not\rightarrow$$

for $1 \leq i \leq n$, where $\rho_i' \neq \mathbf{1}$ and $\boldsymbol{\delta}_i = \boldsymbol{\gamma}_i = [\,]$ by Lemma 3.7. This implies

$$\textstyle\sum_{i\in I\setminus\{k_1\}} \alpha_i.\rho_i \prec \rho_{k_1} \parallel \sum_{j\in J\setminus\{k_1\}} \overline{\alpha}_j.\sigma_j \prec \sigma_{k_1} \xrightarrow{*} \sum_{i\in I\setminus\{k_1\}} \alpha_i.\rho_i \prec \rho_1' \parallel \sum_{j\in J\setminus\{k_1\}} \overline{\alpha}_j.\sigma_j \prec \sigma_1'$$

Let $I' = I \setminus J$ and $J' = J \setminus I$. We can reduce $[\,]\prec \rho \parallel [\,]\prec \sigma$ only as follows:

$$\begin{array}{rcll}
[\,]\prec \rho \parallel [\,]\prec \sigma & \longrightarrow & \sum_{i\in I\setminus\{k_1\}} \alpha_i.\rho_i \prec \rho_{k_1} \parallel \sum_{j\in J\setminus\{k_1\}} \overline{\alpha}_j.\sigma_j \prec \sigma_{k_1} & \text{by } (\mathsf{comm}) \\
& \xrightarrow{*} & \sum_{i\in I\setminus\{k_1\}} \alpha_i.\rho_i \prec \rho_1' \parallel \sum_{j\in J\setminus\{k_1\}} \overline{\alpha}_j.\sigma_j \prec \sigma_1' & \\
& \longrightarrow & [\,]\prec \sum_{i\in I\setminus\{k_1\}} \alpha_i.\rho_i \parallel [\,]\prec \sum_{j\in J\setminus\{k_1\}} \overline{\alpha}_j.\sigma_j & \text{by } (\mathsf{rbk}) \\
& \vdots & \vdots & \\
& \xrightarrow{*} & \sum_{i\in I'} \alpha_i.\rho_i \prec \rho_n' \parallel \sum_{j\in J'} \overline{\alpha}_j.\sigma_j \prec \sigma_n' & \\
& \longrightarrow & [\,]\prec \sum_{i\in I'} \alpha_i.\rho_i \parallel [\,]\prec \sum_{j\in J'} \overline{\alpha}_j.\sigma_j & \text{by } (\mathsf{rbk})
\end{array}$$

and $[\,]\prec \sum_{i\in I'} \alpha_i.\rho_i \parallel [\,]\prec \sum_{j\in J'} \overline{\alpha}_j.\sigma_j$ is stuck since $I' \cap J' = \emptyset$.

Suppose $\rho = \bigoplus_{i\in I} \overline{\alpha}_i.\rho_i$ and $\sigma = \sum_{j\in J} \alpha_j.\sigma_j$. If $I \not\subseteq J$ let $k \in I \setminus J$; then we get

$$[\,]\prec \rho \parallel [\,]\prec \sigma \quad \begin{array}{c}\longrightarrow \\ \not\rightarrow\end{array} \quad [\,]\prec \overline{\alpha}_k.\rho_k \parallel [\,]\prec \sigma \quad \text{by } (\tau)$$

Otherwise $I \subseteq J$ and $\rho_k \not\dashv^{\mathsf{rbk}} \sigma_k$ for some $k \in I$. By reasoning as above we have

$$[\,]\prec \rho_k \parallel [\,]\prec \sigma_k \xrightarrow{*} [\,]\prec \rho' \parallel [\,]\prec \sigma' \not\rightarrow$$

and

$$\circ\prec \rho_k \parallel \textstyle\sum_{j\in J\setminus\{k\}} \alpha_j.\sigma_j \prec \sigma_k \xrightarrow{*} \circ\prec \rho' \parallel \sum_{j\in J\setminus\{k\}} \alpha_j.\sigma_j \prec \sigma'$$

which imply

$$\begin{array}{rcll}
[\,]\prec \rho \parallel [\,]\prec \sigma & \longrightarrow & [\,]\prec \overline{\alpha}_k.\rho_k \parallel [\,]\prec \sigma & \text{by } (\tau) \\
& \longrightarrow & \circ\prec \rho_k \parallel \sum_{j\in J\setminus\{k\}} \alpha_j.\sigma_j \prec \sigma_k & \text{by } (\mathsf{comm}) \\
& \xrightarrow{*} & \circ\prec \rho' \parallel \sum_{j\in J\setminus\{k\}} \alpha_j.\sigma_j \prec \sigma' & \\
& \longrightarrow & [\,]\prec \circ \parallel [\,]\prec \sum_{j\in J\setminus\{k\}} \alpha_j.\sigma_j & \text{by } (\mathsf{rbk}) \\
& \not\rightarrow & &
\end{array}$$

[8] This proof of the direction of the Lemma was presented in the workshop paper [4]. We restate it here for the reader's convenience.

In both cases we conclude that $\rho \not\dashv^{\mathsf{rtk}} \sigma$.
The proof in case $\rho = \sum_{i\in I} \overline{\alpha}_i.\rho_i$, $\sigma = \bigoplus_{j\in J} \alpha_j.\sigma_j$ is similar. □

From Lemma B.1 and definition of $\mathbb{AC}$ we immediately have $\dashv^{\mathsf{rtk}} = \mathbb{AC}$.

B.3. **Proof of** (1) ⇔ (3) **of Theorem 5.10.** Since (by Theorem 2.9) we have that $\mathbb{AC} = \dashv_{\mathsf{tb}}$, it is enough to show that

$$\rho \dashv_{\mathsf{tb}} \sigma \text{ if and only if there exists a winning strategy for player } \mathsf{C} \text{ in } \mathcal{G}_{\rho \parallel \sigma} \tag{B.1}$$

We recall that a strategy Σ is *univocal* if $|\Sigma(\boldsymbol{e})| \leq 1$ for all $\boldsymbol{e}$.
We begin by (B.1)(⇐)

Definition B.2.

(1) Let $\tilde{\rho}, \tilde{\sigma} \in \mathsf{ASC}^{[]}$, let Σ be a univocal strategy for player C and let $\boldsymbol{e}$ be a play. We define the tree $\mathsf{rt\text{-}aux}_\Sigma(\tilde{\rho} \parallel\!\parallel \tilde{\sigma}, \boldsymbol{e})$ as follows:

$$\mathsf{rt\text{-}aux}_\Sigma(\mathbf{1} \parallel\!\parallel \tilde{\sigma}, \boldsymbol{e}) = \begin{array}{c} \mathbf{1} \parallel\!\parallel \tilde{\sigma} \\ | \\ \mathbf{0} \parallel\!\parallel \tilde{\sigma} \end{array}$$

$$\mathsf{rt\text{-}aux}_\Sigma(\tilde{\rho} \parallel\!\parallel \tilde{\sigma}, \boldsymbol{e}) = \begin{array}{c} \tilde{\rho} \parallel\!\parallel \tilde{\sigma} \\ /\cdots\backslash \\ \mathsf{T}_1 \cdots\ \mathsf{T}_n \end{array}$$

if $\exists \beta \in B.\, \tilde{\rho} \parallel\!\parallel \tilde{\sigma} \xrightarrow{\beta}\!\!\!\!\rightarrow$
where $\mathsf{T}_1 = \mathsf{rt\text{-}aux}_\Sigma(\tilde{\rho}_1 \parallel\!\parallel \tilde{\sigma}_1, \boldsymbol{e}\beta_1), \ldots, \mathsf{T}_n = \mathsf{rt\text{-}aux}_\Sigma(\tilde{\rho}_n \parallel\!\parallel \tilde{\sigma}_n, \boldsymbol{e}\beta_n)$
with $\{\, \tilde{\rho}_i \parallel\!\parallel \tilde{\sigma}_i \,\}_{i=1..n} = \{\, \tilde{\rho}' \parallel\!\parallel \tilde{\sigma}' \mid \tilde{\rho} \parallel\!\parallel \tilde{\sigma} \xrightarrow{\beta}\!\!\!\!\rightarrow \tilde{\rho}' \parallel\!\parallel \tilde{\sigma}',\ \beta \in B \,\}$
and $\tilde{\rho} \parallel\!\parallel \tilde{\sigma} \xrightarrow{\beta_i}\!\!\!\!\rightarrow \tilde{\rho}_i \parallel\!\parallel \tilde{\sigma}_i$

$$\mathsf{rt\text{-}aux}_\Sigma(\tilde{\rho} \parallel\!\parallel \tilde{\sigma}, \boldsymbol{e}) = \begin{array}{c} \tilde{\rho} \parallel\!\parallel \tilde{\sigma} \\ | \\ \mathsf{T} \end{array}$$

if $\tilde{\rho} \parallel\!\parallel \tilde{\sigma} \xrightarrow{\mathsf{C}:a}\!\!\!\!\rightarrow \tilde{\rho}' \parallel\!\parallel \tilde{\sigma}'$
where $\Sigma(\boldsymbol{e}) = (k, \mathsf{C}{:}a)$ for some k ,
and where $\mathsf{T} = \mathsf{rt\text{-}aux}_\Sigma(\tilde{\rho}' \parallel\!\parallel \tilde{\sigma}', \boldsymbol{e}\Sigma(\boldsymbol{e}))$

$$\mathsf{rt\text{-}aux}_\Sigma(\tilde{\rho} \parallel\!\parallel \tilde{\sigma}, \boldsymbol{e}) = \begin{array}{c} \tilde{\rho} \parallel\!\parallel \tilde{\sigma} \\ | \\ \boldsymbol{\times} \end{array} \quad \text{otherwise}$$

(2) Let $\rho, \sigma \in \mathsf{ASC}$ and let Σ be a univocal strategy for player C for the game $\mathcal{G}_{\rho\parallel\sigma}$. We define

$$\mathsf{rt}_\Sigma(\rho \parallel\!\parallel \sigma) \triangleq \mathsf{rt\text{-}aux}_\Sigma(\rho \parallel\!\parallel \sigma, \varepsilon)$$

Lemma B.3. *Let Σ be a univocal winning strategy for player C for the game $\mathcal{G}_{\rho\parallel\sigma}$. Then all the leaves of $\mathsf{rt}_\Sigma(\rho \parallel\!\parallel \sigma)$ are of the form $\mathbf{0} \parallel\!\parallel \tilde{\sigma}$.*

We hence get (B.1)(⇐) by Lemmas B.3 and A.4.

We can now proceed with (B.1)(⇒).

Definition B.4. Let $\mathsf{T} \in \mathsf{rts}(\rho \| \sigma)$ such that all its leaves are of the form $\mathbf{0} \parallel \tilde{\sigma}$.

(1) The moves-labelled tree of T, dubbed $\mathsf{mlt}(\mathsf{T})$ is obtained out of T by labelling its edges as follows: if E is an edge from a node N to a child M of its, we label it by $(L(N), \beta)$, where β is such that $N \xrightarrow{\beta} M$ and $L(N)$ is the level of N.
(2) Given a finite path $\boldsymbol{p}$ in $\mathsf{mlt}(\mathsf{T})$ starting from the root, we define $\textsc{fp}(\boldsymbol{p})$ as the sequence of labels of the edges of the path.
(3) Given a finite path $\boldsymbol{p}$ in $\mathsf{mlt}(\mathsf{T})$ starting from the root, we define

$$\mathsf{nextm}(\boldsymbol{p}) \triangleq \begin{cases} (n, \beta) & if\ (*) \\ \emptyset & \text{otherwise} \end{cases}$$

(*) the last node N in $\boldsymbol{p}$ is of the form $\sum_{i \in I} \alpha_i.\tilde{\rho}_i \parallel \sum_{j \in J} \overline{\alpha}_j.\tilde{\sigma}_j$ or $\mathbf{1} \parallel \tilde{\sigma}$ and (n, β) is the label of the only edge out of N.

Lemma B.5. *Let $\mathsf{T} \in \mathsf{rts}(\rho \| \sigma)$ such that all its leaves are of the form $\mathbf{0} \parallel \tilde{\sigma}$ and let $\boldsymbol{p}$ be a finite path in $\mathsf{mlt}(\mathsf{T})$. Then $\textsc{fp}(\boldsymbol{p})$ is a finite play of $\mathcal{G}_{\rho \| \sigma}$*

Proof. By Definition B.4 and by definition of play of $\mathcal{G}_{\rho \| \sigma}$. □

Definition B.6. Let $\mathsf{T} \in \mathsf{rts}(\rho \| \sigma)$ such that all its leaves are of the form $\mathbf{0} \parallel \tilde{\sigma}$. We define the strategy Σ_{T} in $\mathcal{G}_{\rho \| \sigma}$, for player C, by

$$\Sigma_{\mathsf{T}}(\boldsymbol{e}) \triangleq \begin{cases} \mathsf{nextm}(\boldsymbol{p}) & if\ (*) \\ \emptyset & \text{otherwise} \end{cases}$$

(*) $\boldsymbol{e} = \textsc{fp}(\boldsymbol{p})$ for some $\boldsymbol{p}$ which is a finite path in $\mathsf{mlt}(\mathsf{T})$.

Lemma B.7. *Let $\mathsf{T} \in \mathsf{rts}(\rho \| \sigma)$ such that all its leaves are of the form $\mathbf{0} \parallel \tilde{\sigma}$. Then Σ_{T} is a univocal winning strategy for player C for the game $\mathcal{G}_{\rho \| \sigma}$.*

We hence get (B.1)(⇒) from the above Lemma and A.4.

B.4. **Proof of** (1) ⇔ (4) **of Theorem 5.10.** In the following proof we could have proceed by using a lemma similar to B.1. However, in order to get also a correspondence between orchestrators and derivations to be used for the proof of the Main Theorem II (5.11), we proceed by providing two formal systems axiomatizing the relation of orchestrated compliance.

B.4.1. *(Formal systems and synthesis for $\dashv^{\mathsf{Orch}}$).* In the following, orchestrators are considered as explicit recursive terms rather than as possibly infinite regular trees. We first define a formal system $\rhd_{\mathsf{o}}$ in which the relation of derivability characterizes the relation $\dashv^{\mathsf{Orch}}$. In System $\rhd_{\mathsf{o}}$ the relation $\dashv^{\mathsf{Orch}}$ is the intended interpretation of the symbol $\backsim^{\mathsf{Orch}}$.

Definition B.8 (Formal System for Orchestrated Compliance)**.** An environment Γ is a finite set of expressions of the form $f : \delta \backsim^{\mathsf{Orch}} \gamma$ where $\delta, \gamma \in \mathsf{ASC}$ and $f \in \mathsf{Orch}$. The judgments of System $\rhd_{\mathsf{o}}$ are expressions of the form $\Gamma \rhd f : \rho \backsim^{\mathsf{Orch}} \sigma$. The axioms and rules of $\rhd_{\mathsf{o}}$ are as in Figure 6.

$$(\text{Ax}) : \overline{\Gamma \rhd_{\mathsf{o}} \mathfrak{1} : \mathbf{1} \backsim^{\mathsf{Orch}} \sigma} \qquad (\text{Hyp}) : \overline{\Gamma, f : \rho \backsim^{\mathsf{Orch}} \sigma \rhd_{\mathsf{o}} f : \rho \backsim^{\mathsf{Orch}} \sigma}$$

$$(+ \cdot +) : \frac{\Gamma' \rhd_{\mathsf{o}} f : \rho \backsim^{\mathsf{Orch}} \sigma}{\Gamma \rhd_{\mathsf{o}} \langle \overline{\alpha}, \alpha \rangle^{+}.f : \alpha.\rho + \rho' \backsim^{\mathsf{Orch}} \overline{\alpha}.\sigma + \sigma'}$$
$$\text{where } \Gamma' = \Gamma, \langle \overline{\alpha}, \alpha \rangle^{+}.f : \alpha.\rho + \rho' \backsim^{\mathsf{Orch}} \overline{\alpha}.\sigma + \sigma'$$

$$(\oplus \cdot +) : \frac{\forall i \in I.\ \Gamma' \rhd_{\mathsf{o}} f_i : \rho_i \backsim^{\mathsf{Orch}} \sigma_i}{\Gamma \rhd_{\mathsf{o}} \bigvee_{i \in \{\, 1..n \,\}} \langle a_i, \overline{a}_i \rangle.f_{a_i} : \bigoplus_{i \in I} \overline{a}_i.\rho_i \backsim^{\mathsf{Orch}} \sum_{j \in I \cup J} a_j.\sigma_j}$$
$$\text{where } \Gamma' = \Gamma, \bigvee_{i \in \{\, i=1..n \,\}} \langle a_i, \overline{a}_i \rangle.f_i : \bigoplus_{i \in I} \overline{a}_i.\rho_i \backsim^{\mathsf{Orch}} \sum_{j \in I \cup J} a_j.\sigma_j$$

$$(+ \cdot \oplus) : \frac{\forall i \in I.\ \Gamma \rhd_{\mathsf{o}} f_i : \rho_i \backsim^{\mathsf{Orch}} \sigma_i}{\Gamma \rhd_{\mathsf{o}} \bigvee_{i \in \{\, i=1..n \,\}} \langle \overline{a}_i, a_i \rangle.f_i : \sum_{j \in I \cup J} a_j.\sigma_j \backsim^{\mathsf{Orch}} \bigoplus_{i \in I} \overline{a}_i.\rho_i}$$
$$\text{where } \Gamma' = \Gamma, \bigvee_{i \in \{\, 1..n \,\}} \langle \overline{a}_i, a_i \rangle.f_i : \sum_{j \in I \cup J} a_j.\sigma_j \backsim^{\mathsf{Orch}} \bigoplus_{i \in I} \overline{a}_i.\rho_i$$

Figure 6: System $\rhd_{\mathsf{o}}$

Theorem B.9 (Soundness and Completeness of System $\rhd_{\mathsf{o}}$ w.r.t $\dashv^{\mathsf{Orch}}$)**.**

$$\rhd_{\mathsf{o}} f : \rho \backsim^{\mathsf{Orch}} \sigma \ \Leftrightarrow \ f : \rho \dashv^{\mathsf{Orch}} \sigma$$

Proof. The proof can be developed along the very same lines of the proofs for Proposition 5.6 and Lemma 5.7 for what concerns, respectively, soundness and completeness. □

We provide now a formal system $\rhd_{\mathsf{o}}^{\mathsf{x}}$ equivalent to $\rhd_{\mathsf{o}}$. In such a system, unproper (namely open) orchestrators can be used. However we shall apply the system only for proper (namely closed) orchestrators.

The algorithm **Synth** corresponds to a proof-search algorithm in system $\rhd_{\mathsf{o}}^{\mathsf{x}}$: it synthesises, given ρ and σ, an orchestrator f such that $f : \rho \dashv^{\mathsf{Orch}} \sigma$, and hence a univocal winning strategy for player C in the game $\mathcal{G}_{\rho \| \sigma}$.

Definition B.10 (The equivalent System $\rhd_{\mathsf{o}}^{\mathsf{x}}$)**.** An environment Γ is a finite set of expressions of the form $x : \delta \backsim^{\mathsf{Orch}} \gamma$ where $\delta, \gamma \in \mathsf{ASC}$ and x is an orchestrator variable. The judgments of System $\rhd_{\mathsf{o}}$ are expressions of the form $\Gamma \rhd f : \rho \backsim^{\mathsf{Orch}} \sigma$, where f is an orchestrator, possibly open. The axioms and rules of $\rhd_{\mathsf{o}}^{\mathsf{x}}$ are as in Figure 7.

Proposition B.11. *Let f be a proper (closed) orchestrator.*

$$\rhd_{\mathsf{o}} f : \rho \backsim^{\mathsf{Orch}} \sigma \quad \Leftrightarrow \quad \rhd_{\mathsf{o}}^{\mathsf{x}} f : \rho \backsim^{\mathsf{Orch}} \sigma$$

Now we show how to build an orchestrator f and a derivation in $\rhd_{\mathsf{o}}^{\mathsf{x}}$ such that $\rhd_{\mathsf{o}}^{\mathsf{x}} f : \rho \backsim \sigma$ when a derivation of $\rhd \rho \backsim \sigma$ is given.

Definition B.12.

(1) The partial functions $\mathsf{fder\text{-}aux}(\text{-},\text{-})$ and $\mathsf{f\text{-}aux}(\text{-},\text{-})$, from derivations in $\rhd$ and environments to, respectively, derivations in $\rhd_{\mathsf{o}}^{\mathsf{x}}$ and (possibly open) orchestrators, are inductively and simultaneously defined as follows:

$$(\text{Ax}) : \overline{\Gamma \rhd_{\mathsf{o}}^{\mathsf{x}} \mathtt{1} : \mathbf{1} \rightharpoonup^{\mathsf{Orch}} \sigma} \qquad (\text{Hyp}) : \overline{\Gamma, x : \rho \rightharpoonup^{\mathsf{Orch}} \sigma \rhd_{\mathsf{o}}^{\mathsf{x}} x : \rho \rightharpoonup^{\mathsf{Orch}} \sigma}$$

$$(+ \cdot +) : \frac{\Gamma' \rhd_{\mathsf{o}}^{\mathsf{x}} f : \rho \rightharpoonup^{\mathsf{Orch}} \sigma}{\Gamma \rhd_{\mathsf{o}}^{\mathsf{x}} \mathsf{rec}\, x.\langle \overline{\alpha}, \alpha \rangle.f : \alpha.\rho + \rho' \rightharpoonup^{\mathsf{Orch}} \overline{\alpha}.\sigma + \sigma'}$$
where $\Gamma' = \Gamma, x : \alpha.\rho + \rho' \rightharpoonup^{\mathsf{Orch}} \overline{\alpha}.\sigma + \sigma'$

$$(\oplus \cdot +) : \frac{\forall i \in I.\ \Gamma' \rhd_{\mathsf{o}}^{\mathsf{x}} f_{a_i} : \rho_i \rightharpoonup^{\mathsf{Orch}} \sigma_i}{\Gamma \rhd_{\mathsf{o}}^{\mathsf{x}} \mathsf{rec}\, x. \bigvee_{\{a_i | i=1..n\}} f_{a_i} : \bigoplus_{i \in I} \overline{a}_i.\rho_i \rightharpoonup^{\mathsf{Orch}} \sum_{j \in I \cup J} a_j.\sigma_j}$$
where $\Gamma' = \Gamma, x : \bigoplus_{i \in I} \overline{a}_i.\rho_i \rightharpoonup^{\mathsf{Orch}} \sum_{j \in I \cup J} a_j.\sigma_j$

$$(+ \cdot \oplus) : \frac{\forall i \in I.\ \Gamma \rhd_{\mathsf{o}}^{\mathsf{x}} f_{a_i} : \rho_i \rightharpoonup^{\mathsf{Orch}} \sigma_i}{\Gamma \rhd_{\mathsf{o}}^{\mathsf{x}} \mathsf{rec}\, x. \bigvee_{\{a_i | i=1..n\}} f_{a_i} : \sum_{j \in I \cup J} a_j.\sigma_j \rightharpoonup^{\mathsf{Orch}} \bigoplus_{i \in I} \overline{a}_i.\rho_i}$$
where $\Gamma' = \Gamma, \bigvee_{\{a_i | i=1..n\}} f_{a_i} : \sum_{j \in I \cup J} a_j.\sigma_j \rightharpoonup^{\mathsf{Orch}} \bigoplus_{i \in I} \overline{a}_i.\rho_i$

Figure 7: System $\rhd_{\mathsf{o}}^{\mathsf{x}}$

$\mathcal{D} = \overline{\Gamma \rhd \mathbf{1} \rightharpoonup \sigma}$ (Ax)**:**

$$\begin{array}{lcl} \mathsf{fder\text{-}aux}(\mathcal{D}, \Gamma^{\mathsf{x}}) & = & \overline{\Gamma^{\mathsf{x}} \rhd_{\mathsf{o}}^{\mathsf{x}} \mathtt{1} : \mathbf{1} \rightharpoonup^{\mathsf{Orch}} \sigma}\ (\text{Ax}) \\ \mathsf{f\text{-}aux}(\mathcal{D}, \Gamma^{\mathsf{x}}) & = & \mathtt{1} \end{array}$$

$\mathcal{D} = \overline{\Gamma, \rho \rightharpoonup \sigma \rhd \rho \rightharpoonup \sigma}$ (Hyp)**:**

$$\left.\begin{array}{lcl} \mathsf{fder\text{-}aux}(\mathcal{D}, \Gamma^{\mathsf{x}}) & = & \overline{\Gamma^{\mathsf{x}} \rhd_{\mathsf{o}}^{\mathsf{x}} x : \rho \rightharpoonup^{\mathsf{Orch}} \sigma}\ (\text{Hyp}) \\ \mathsf{f\text{-}aux}(\mathcal{D}, \Gamma^{\mathsf{x}}) & = & x \end{array}\right\} \text{ if } x : \rho \rightharpoonup^{\mathsf{Orch}} \sigma \in \Gamma^{\mathsf{x}}.$$

$\mathcal{D} = \dfrac{\mathcal{D}'}{\Gamma \rhd \alpha.\rho + \rho' \rightharpoonup \overline{\alpha}.\sigma + \sigma'}$ $(+ \cdot +)$**:**

$$\begin{array}{lcl} \mathsf{fder\text{-}aux}(\mathcal{D}, \Gamma^{\mathsf{x}}) & = & \dfrac{\mathsf{fder\text{-}aux}(\mathcal{D}', (\Gamma^{\mathsf{x}}, x{:}\alpha.\rho + \rho' \rightharpoonup^{\mathsf{Orch}} \overline{\alpha}.\sigma + \sigma') = \Gamma^{\mathsf{x}\prime})}{\Gamma^{\mathsf{x}} \rhd_{\mathsf{o}}^{\mathsf{x}} \mathsf{rec}\, x.\langle \overline{\alpha}, \alpha \rangle.\mathsf{f\text{-}aux}(\mathcal{D}', \Gamma^{\mathsf{x}\prime}) : \alpha.\rho + \rho' \rightharpoonup^{\mathsf{Orch}} \overline{\alpha}.\sigma + \sigma'}\ (+ \cdot +) \\ \mathsf{f\text{-}aux}(\mathcal{D}, \Gamma^{\mathsf{x}}) & = & \mathsf{rec}\, x.\langle \overline{\alpha}, \alpha \rangle.\mathsf{f\text{-}aux}(\mathcal{D}', \Gamma^{\mathsf{x}\prime}) \\ & & \text{where } x \text{ is a fresh variable.} \end{array}$$

$\mathcal{D} = \dfrac{\forall i \in I.\ \mathcal{D}_i}{\Gamma \rhd \bigoplus_{i \in I} \overline{a}_i.\rho_i \rightharpoonup \sum_{j \in I \cup J} a_j.\sigma_j}$ $(\oplus \cdot +)$**:**

$$\begin{array}{lcl} \mathsf{fder\text{-}aux}(\mathcal{D}, \Gamma^{\mathsf{x}}) & = & \dfrac{\forall i \in I.\ \mathsf{fder\text{-}aux}(\mathcal{D}_i, (\Gamma^{\mathsf{x}}, x{:}\bigoplus_{i \in I} \overline{a}_i.\rho_i \rightharpoonup^{\mathsf{Orch}} \sum_{j \in I \cup J} a_j.\sigma_j) = \Gamma^{\mathsf{x}\prime})}{\Gamma \rhd_{\mathsf{o}}^{\mathsf{x}} \mathsf{rec}\, x. \bigvee_{\{a_i | i=1..n\}} f_{a_i} : \bigoplus_{i \in I} \overline{a}_i.\rho_i \rightharpoonup^{\mathsf{Orch}} \sum_{j \in I \cup J} a_j.\sigma_j}\ (\oplus \cdot +) \\ \mathsf{f\text{-}aux}(\mathcal{D}, \Gamma^{\mathsf{x}}) & = & \mathsf{rec}\, x. \bigvee_{\{a_i | i=1..n\}} \mathsf{f\text{-}aux}(\mathcal{D}_i, \Gamma^{\mathsf{x}\prime}) \\ & & \text{where } f_{a_i} = \mathsf{f\text{-}aux}(\mathcal{D}_i, \Gamma^{\mathsf{x}\prime}) \text{ and } x \text{ is a fresh variable.} \end{array}$$

$\mathcal{D} = \dfrac{\forall i \in I.\ \mathcal{D}_i}{\Gamma \rhd \sum_{j \in I \cup J} a_j.\sigma_j \rightharpoonup \bigoplus_{i \in I} \overline{a}_i.\rho_i}(+ \cdot \oplus)$**:**

$$\begin{array}{lcl}\mathsf{fder\text{-}aux}(\mathcal{D},\Gamma^{\mathsf{x}}) & = & \dfrac{\forall i\in I.\ \mathsf{fder\text{-}aux}(\mathcal{D}_i,(\Gamma^{\mathsf{x}},x{:}\sum_{j\in I\cup J}a_j.\sigma_j \multimap^{\mathsf{Orch}}\bigoplus_{i\in I}\overline{a}_i.\rho_i)=\Gamma^{\mathsf{x}\prime})}{\Gamma\rhd^{\mathsf{x}}_{\mathsf{o}}\ \mathsf{rec}\,x.\bigvee_{\{\,a_i|i=1..n\,\}}f_{a_i}:\sum_{j\in I\cup J}a_j.\sigma_j\multimap^{\mathsf{Orch}}\bigoplus_{i\in I}\overline{a}_i.\rho_i}\\ \mathsf{f\text{-}aux}(\mathcal{D},\Gamma^{\mathsf{x}}) & = & \mathsf{rec}\,x.\bigvee_{\{\,a_i|i=1..n\,\}}\mathsf{f\text{-}aux}(\mathcal{D}_i,\Gamma^{\mathsf{x}\prime})\\ & & \text{where } f_{a_i}=\mathsf{f\text{-}aux}(\mathcal{D}_i,\Gamma^{\mathsf{x}\prime}) \text{ and } x \text{ is a fresh variable.}\end{array}$$

(2) The partial function $\mathsf{fder}(\text{-})$ from derivations in $\rhd$ to derivations in $\rhd^{\mathsf{x}}_{\mathsf{o}}$, and the partial function $\mathsf{f\text{-}aux}(\text{-})$ from derivations in $\rhd$ to (possibly open) orchestrators are defined by

$$\mathsf{fder}(\mathcal{D})=\mathsf{fder\text{-}aux}(\mathcal{D},\emptyset)$$
$$\mathsf{f}(\mathcal{D})=\mathsf{f\text{-}aux}(\mathcal{D},\emptyset)$$

Lemma B.13. *Let $\mathcal{D}::\rhd\ \rho\multimap\sigma$. Then $\mathsf{fder}(\mathcal{D})$ and $\mathsf{f}(\mathcal{D})$ are well-defined, $\mathsf{f}(\mathcal{D})$ is a proper (i.e. closed) orchestrator and*

$$\mathsf{fder}(\mathcal{D})::\rhd^{\mathsf{x}}_{\mathsf{o}}\ \mathsf{f}(\mathcal{D}):\ \rho\multimap^{\mathsf{Orch}}\sigma$$

Proof. By Induction. □

We hence get (1) ⇒ (4) as an immediate consequence of Lemma B.13 above, Proposition B.11 and Theorem B.9.

The implication (4) ⇒ (1) is instead an easy consequence of the observation that by erasing all orchestrators in a derivation of $\rhd_{\mathsf{o}}\ f:\rho\multimap\sigma$ we get a derivation of $\rhd\ \rho\multimap\sigma$.

Appendix C. Proof of Main Theorem II (5.11) (Getting derivations, orchestrators and strategies out of each other)

We begin by providing a stratified version of orchestrated compliance.

Definition C.1 (Coinductive orchestrated compliance)**.**
Let $\{\dashv^{\mathsf{Orch}}_k\}_{k\in\mathbb{N}}$ be the family of relations over $\mathsf{Orch}\times\mathsf{ASC}\times\mathsf{ASC}$ such that
(1) $\dashv^{\mathsf{Orch}}_0=\mathsf{Orch}\times\mathsf{ASC}\times\mathsf{ASC}$ and
(2) $f:\rho\ \dashv^{\mathsf{Orch}}_{k+1}\ \sigma$ if either:
  (a) $\rho=\mathbf{1}$; or
  (b) $\rho\neq\mathbf{1}$, $\rho\parallel_f\sigma\Longrightarrow$, and $\rho\parallel_f\sigma\Longrightarrow\rho'\parallel_{f'}\sigma'$ implies $f':\rho'\ \dashv^{\mathsf{Orch}}_k\ \sigma'$.
Then we define $\dashv^{\mathsf{Orch}}_{co}=\bigcap_{k\in\mathbb{N}}\dashv^{\mathsf{Orch}}_k$.

Proposition C.2. *The relation $\dashv^{\mathsf{Orch}}_{co}$ and the compliance relation $\dashv^{\mathsf{Orch}}$ coincide, i.e.*

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

As done for the relation $\mathbb{AC}$, we provide an equivalent "turn-based" version of the relation $\dashv^{\mathsf{Orch}}$.

Definition C.3 (Turn-based operational semantics of turn-based orchestrated configurations)**.** Let $\mathbf{tbAct}=\{\mathsf{A},\mathsf{B},\mathsf{C}\}\times(\mathbf{Act}\cup\{\checkmark\})$. In Figure 8 we define the LTS $\longrightarrow\!\!\!\!\!\rightarrow_o$ over turn-based configurations, with labels in $\mathbf{tbAct}$.

We define $\longrightarrow\!\!\!\!\!\rightarrow_o=\bigcup_{\beta\in\mathbf{tbAct}}\overset{\beta}{\longrightarrow\!\!\!\!\!\rightarrow}_o$.

Definition C.4 (Turn-based orchestrated compliance $\dashv^{\mathsf{Orch}}_{\mathbf{b}}$)**.** Let $f\in\mathsf{Orch}$ and $\tilde{\rho},\tilde{\sigma}\in\mathsf{ASC}^{[]}$.

$$\oplus_{i\in I}\overline{a}_i.\rho_i \parallel_f \sigma \quad \xrightarrow{\mathsf{A}:\overline{a}_k}_o \quad [\overline{a}_k]\rho_k \parallel_f \sigma \qquad (k \in I)$$

$$\Sigma_{i\in I}a_i.\rho_i \parallel_{\bigvee_{h\in H}\langle\overline{a}_h,a_h\rangle.f_h} [\overline{a}_k]\sigma \quad \xrightarrow{\mathsf{A}:a_k}_o \quad \rho_k \parallel_{f_k} \sigma \qquad (k \in (I \cap H))$$

$$\rho \parallel_f \oplus_{i\in I}\overline{a}_i.\sigma_i \quad \xrightarrow{\mathsf{B}:\overline{a}_k}_o \quad \rho \parallel_f [\overline{a}_k]\sigma_k \qquad (k \in I)$$

$$[\overline{a}_k]\rho \parallel_{\bigvee_{h\in H}\langle a_h,\overline{a}_h\rangle.f_h} \Sigma_{i\in I}a_i.\sigma_i \quad \xrightarrow{\mathsf{B}:a_k}_o \quad \rho \parallel_{f_k} \sigma_k \qquad (k \in (I \cap H))$$

$$\overline{a}.\rho + \rho' \parallel_{\langle a,\overline{a}\rangle^+.f'} a.\sigma + \sigma' \quad \xrightarrow{\mathsf{C}:a}_o \quad \rho \parallel_{f'} \sigma \qquad a.\rho + \rho' \parallel_{\langle\overline{a},a\rangle^+.f'} \overline{a}.\sigma + \sigma' \quad \xrightarrow{\mathsf{C}:a}_o \quad \rho \parallel_{f'} \sigma$$

$$\mathbf{1} \parallel_f \tilde{\rho} \quad \xrightarrow{\mathsf{C}:\checkmark}_o \quad \mathbf{0} \parallel_f \tilde{\rho}$$

Figure 8: Turn-based operational semantics of orchestrated-configurations systems

(1) $f : \tilde{\rho} \dashv\!\!\!\dashv^{\mathsf{Orch}}_{\mathfrak{b}} \tilde{\sigma}$ if

$$\tilde{\rho} \parallel_f \tilde{\sigma} \;\longrightarrow\!\!\!\!\!\rightarrow_o^* \; \tilde{\rho}' \parallel_{f'} \tilde{\sigma}' \not\longrightarrow\!\!\!\!\!\rightarrow_o \quad \text{implies} \quad \tilde{\rho}' = \mathbf{1}.$$

(2) $\tilde{\rho} \dashv\!\!\!\dashv^{\mathsf{Orch}}_{\mathfrak{b}} \tilde{\sigma}$ if $\exists f.\, [\, f : \tilde{\rho} \dashv\!\!\!\dashv^{\mathsf{Orch}}_{\mathfrak{b}} \tilde{\sigma}\,]$.

Along the very same lines of the proof of Theorem 2.9, it is possible to show the equivalence of $\dashv^{\mathsf{Orch}}$ and $\dashv^{\mathsf{Orch}}_{\mathfrak{b}}$.

Lemma C.5. *Let $\rho, \sigma \in \mathsf{ASC}(\subseteq \mathsf{ASC}^{[]})$ and $f \in \mathsf{Orch}$.*

$$f : \rho \dashv^{\mathsf{Orch}} \sigma \;\Leftrightarrow\; f : \rho \dashv^{\mathsf{Orch}}_{\mathfrak{b}} \sigma$$

C.1. **Proof of Theorem 5.11(1).** We can simply use the function $\mathsf{f}(\text{-}, \emptyset)$ described in the proof of Lemma B.13.

C.2. **Proof of Theorem 5.11(2).** We proceed as follows: given an orchestrator f such that $f : \rho \dashv^{\mathsf{Orch}}_{\mathfrak{b}} \sigma$ (and hence $f : \rho \dashv^{\mathsf{Orch}} \sigma$, by Lemma C.5) we build a regular tree which corresponds to a tree in $\mathsf{rts}(\rho \parallel \sigma)$ (see Def. A.3) with no leaf of the form ✗. We can then decorate such a tree so that it is easy to obtain a winning strategy for player C in $\mathcal{G}_{\rho\|\sigma}$.

We begin by showing how to get a regular tree out of a turn-based orchestrated system

Definition C.6. Let $\tilde{\rho}, \tilde{\sigma} \in \mathsf{ASC}^{[]}$ and $f \in \mathsf{Orch}$. We define the regular tree of the orchestrated system $\tilde{\rho} \parallel_f \tilde{\sigma}$, which we dub $\mathsf{rt}(\tilde{\rho} \parallel_f \tilde{\sigma})$, as follows:

let $B = \{\mathsf{A}:a, \mathsf{A}:\overline{a}, \mathsf{B}:a, \mathsf{B}:\overline{a} \mid a \in \mathcal{N}\}$

$$\mathsf{rt}(\mathbf{1} \parallel_f \tilde{\sigma}) = \begin{array}{c} \mathbf{1} \parallel_f \tilde{\sigma} \\ | \\ \mathbf{0} \parallel_f \tilde{\sigma} \end{array}$$

$$\mathsf{rt}(\tilde{\rho} \parallel_f \tilde{\sigma}) = \begin{array}{c} \tilde{\rho} \parallel_f \tilde{\sigma} \\ / \cdots \backslash \\ \mathsf{T}_1 \cdots \mathsf{T}_n \end{array} \quad \text{if } \exists \beta \in B.\, \tilde{\rho} \parallel_f \tilde{\sigma} \xrightarrow{\beta}\!\!\!\!\rightarrow_o$$

where $\{\, \tilde{\rho}_i \parallel_{f_i} \tilde{\sigma}_i \,\}_{i=1..n} = \{\, \tilde{\rho}' \parallel_f \tilde{\sigma}' \mid \tilde{\rho} \parallel_f \tilde{\sigma} \xrightarrow{\beta}\!\!\!\!\rightarrow_o \tilde{\rho}' \parallel_{f'} \tilde{\sigma}',\ \beta \in B \,\}$
and $\mathsf{T}_i = \mathsf{rt}(\tilde{\rho}_i \parallel_{f_i} \tilde{\sigma}_i)$ $(i = 1..n)$

$$\mathsf{rt}(\tilde{\rho} \parallel_f \tilde{\sigma}) = \begin{array}{c} \tilde{\rho} \parallel_f \tilde{\sigma} \\ | \\ \mathsf{T} \end{array} \quad \text{if } \exists a \in \mathcal{N}.\, \tilde{\rho} \parallel_f \tilde{\sigma} \xrightarrow{\mathsf{C}:a}\!\!\!\!\rightarrow_o \tilde{\rho}' \parallel_{f'} \tilde{\sigma}'$$

where $\mathsf{T} = \mathsf{rt}(\tilde{\rho}' \parallel_{f'} \tilde{\sigma}')$

$$\mathsf{rt}(\tilde{\rho} \parallel_f \tilde{\sigma}) = \begin{array}{c} \tilde{\rho} \parallel_f \tilde{\sigma} \\ | \\ \boldsymbol{\times} \end{array} \quad \text{if } \rho \neq \mathbf{1} \text{ and } \tilde{\rho} \parallel_f \tilde{\sigma} \not\xrightarrow{}\!\!\!\!\rightarrow$$

Notice that the condition of the third clause in the above definition is actually nondeterministic; so, strictly speaking, we are not defining a function. We can get a proper function definition by any method through which it is possible to get rid of such an ambiguity. For instance, by totally ordering the set $\mathcal{N}$ and considering the first element of its satisfying the condition.

Lemma C.7. *Let $\tilde{\rho}, \tilde{\sigma} \in \mathsf{ASC}^{[]}$ and $f \in \mathsf{Orch}$. $f : \tilde{\rho} \dashv\!\!\!\!\flat^{\mathsf{Orch}}\ \tilde{\sigma}$ iff in $\mathsf{rt}(\tilde{\rho} \parallel_f \tilde{\sigma})$ all the leaves are of the form $\mathbf{0} \parallel_f \tilde{\sigma}$*

Given $\mathsf{rt}(\rho \parallel_f \sigma)$, we denote by $\mathsf{rt}(\rho \parallel_f \sigma)^-$ the tree obtained out of $\mathsf{rt}(\rho \parallel_f \sigma)$ by erasing the orchestrators from the label of its nodes.

Lemma C.8. *Let $\rho, \sigma \in \mathsf{ASC}(\subseteq \mathsf{ASC}^{[]})$ and $f \in \mathsf{Orch}$. If $f : \rho \dashv\!\!\!\!\flat^{\mathsf{Orch}}\ \sigma$, then $\mathsf{rt}(\rho \parallel_f \sigma)^- \in \mathsf{rts}(\rho \parallel \sigma)$.*

Definition C.9. Let $\rho, \sigma \in \mathsf{ASC}(\subseteq \mathsf{ASC}^{[]})$ and $f \in \mathsf{Orch}$. We define the strategy regular tree of the orchestrated system $\rho \parallel_f \sigma$, which we dub $\mathsf{srt}(\rho \parallel_f \sigma)$, as follows:
let $B = \{\mathsf{A}:a, \mathsf{A}:\overline{a}, \mathsf{B}:a, \mathsf{B}:\overline{a} \mid a \in \mathcal{N}\}$ and let $\mathsf{srt\text{-}aux}(\tilde{\rho} \parallel_f \tilde{\sigma}, n)$ with $\tilde{\rho}, \tilde{\sigma} \in \mathsf{ASC}$ and

$n \in \mathbb{N}$ be defined by

$$\mathsf{srt\text{-}aux}(\mathbf{1} \parallel_f \tilde{\sigma}, n) \quad = \quad \begin{array}{c} \mathbf{1} \parallel_f \tilde{\sigma} \\ \big| \ (n+1, \mathsf{C} : \checkmark) \\ \mathbf{0} \parallel \tilde{\sigma} \end{array}$$

$$\mathsf{srt\text{-}aux}(\tilde{\rho} \parallel_f \tilde{\sigma}, n) \quad = \quad \begin{array}{c} \tilde{\rho} \parallel_f \tilde{\sigma} \\ (n+1,\beta_1) / \cdots \backslash (n+1,\beta_n) \\ \mathsf{T}_1 \cdots \mathsf{T}_n \end{array} \quad \text{if } \exists \beta \in B.\, \tilde{\rho} \parallel \tilde{\sigma} \xrightarrow{\beta}_o$$

where $\tilde{\rho} \parallel_f \tilde{\sigma} \xrightarrow{\beta_i}_o \tilde{\rho}_i \parallel_{f'} \tilde{\sigma}_i$, $\beta_i \in B$, $i = 1..n$, and $\mathsf{T}_i = \mathsf{srt\text{-}aux}(\tilde{\rho}_i \parallel \tilde{\sigma}_i, n+1)$

$$\mathsf{srt\text{-}aux}(\tilde{\rho} \parallel_f \tilde{\sigma}, n) \quad = \quad \begin{array}{c} \tilde{\rho} \parallel_f \tilde{\sigma} \\ \big| \ (n+1, \mathsf{C}{:}a) \\ \mathsf{T} \end{array} \quad \text{if } \exists a \in \mathcal{N}.\, \tilde{\rho} \parallel_f \tilde{\sigma} \xrightarrow{\mathsf{C}:a}_o \tilde{\rho}' \parallel_{f'} \tilde{\sigma}'$$

where $\mathsf{T} = \mathsf{rt}(\tilde{\rho}' \parallel_{f'} \tilde{\sigma}')$

$$\mathsf{srt\text{-}aux}(\tilde{\rho} \parallel_f \tilde{\sigma}, n) \quad = \quad \begin{array}{c} \tilde{\rho} \parallel_f \tilde{\sigma} \\ \big| \\ \boldsymbol{\times} \end{array} \quad \text{if } \rho \neq \mathbf{1} \text{ and } \tilde{\rho} \parallel \tilde{\sigma} \not\xrightarrow{}_o$$

hence

$$\mathsf{srt}(\rho \parallel_f \sigma) = \mathsf{srt\text{-}aux}(\rho \parallel_f \sigma, 0).$$

The same observation we made after Definition C.6 holds here for what concerns the third clause of the definition of srt-aux.

Notice that, by construction, $\mathsf{srt}(\rho \parallel_f \sigma)$ is but a "decorated" version of $\mathsf{rt}(\rho \parallel \sigma)$. We hence get the following

Fact C.10. All the leaves of $\mathsf{srt}(\rho \parallel_f \sigma)$ are of the form $\mathbf{0} \parallel \tilde{\sigma}$ if and only if all the leaves of $\mathsf{rt}(\rho \parallel \sigma)$ are so.

Definition C.11. Given an orchestrated system $\rho \parallel_f \sigma$, the strategy Σ_f for C in the game $\mathcal{G}_{\rho \parallel \sigma}$ is defined as follows:
let $\boldsymbol{e} = \langle e_0 \cdots e_n \rangle$ be a finite play in $\mathcal{G}_{\rho \parallel \sigma}$

$$\Sigma_f(\boldsymbol{e}) = \begin{cases} (k+1, \mathsf{C}{:}a) & if\ (*) \\ \emptyset & \text{otherwise} \end{cases}$$

(*) $\boldsymbol{e}$ is a sequence of labels in $\mathsf{srt}(\tilde{\rho} \parallel_f \tilde{\sigma})$ from the root to a node of the form $\overline{\alpha}.\rho' + \rho'' \parallel_{f'} \alpha.\sigma' + \sigma''$, and where where (k, β) is the label of the arc above such a node node ($k = 0$ if the node coincides with the root).

Fact C.12. Given an orchestrated system $\rho \parallel_f \sigma$, the strategy Σ_f is univocal.

Lemma C.13. *Let $f : \rho \dashv^{\mathsf{Orch}} \sigma$. The strategy Σ_f is univocal and winning for C in the game $\mathcal{G}_{\rho \parallel \sigma}$.*

Proof. Let $f : \rho \dashv^{\mathsf{Orch}} \sigma$, then by Lemma C.5 we have that $f : \rho \dashv_{\mathbf{b}}^{\mathsf{Orch}} \sigma$. By Lemma C.7, no leaf in $\mathsf{rt}(\rho \parallel_f \sigma)$, and hence in $\mathsf{rt}(\rho \parallel_f \sigma)$ (by Fact C.10), is of the form $\boldsymbol{\times}$. By definition of Σ_f and by Lemma 2.16, it is winning for C in the game $\mathcal{G}_{\rho \parallel \sigma}$. It is also univocal by Fact C.12. □

C.3. **Proof of Theorem 5.11(3).**

Let Σ be a univocal winning strategy for player C for the game $\mathcal{G}_{\rho\|\sigma}$. We then take into account the tree $\mathsf{rt}_\Sigma(\rho \| \sigma)$ as defined in Definition B.2. We show now how to get an orchestrator such that $f : \rho \dashv\!\!\!\flat^{\mathsf{Orch}} \sigma$ (and hence $f : \rho \dashv^{\mathsf{Orch}} \sigma$) out of $\mathsf{rt}_\Sigma(\rho \| \sigma)$.

By means of the following definition we shall be able to get an orchestrator out of a tree in $\mathsf{rts}(\rho \parallel \sigma)$ which does not contain leaves of the form $\boldsymbol{\times}$.

Definition C.14. Let $\mathsf{T} \in \mathsf{rts}(\rho \parallel \sigma)$, the unproper orchestrator $\mathsf{orch}(\mathsf{T})$ (that is possibly containing leaves of the form $\boldsymbol{\times}$) is defined as follows:

$$\mathsf{orch}\Big(\begin{array}{c}\mathbf{1} \parallel \tilde{\sigma}\\ |\\ \mathbf{0} \parallel \tilde{\sigma}\end{array}\Big) \quad = \quad \mathtt{1}$$

$$\mathsf{orch}\Big(\begin{array}{c}\tilde{\rho} \parallel \tilde{\sigma}\\ /\cdots\backslash\\ \mathsf{T}_1 \cdots \mathsf{T}_n\end{array}\Big) \quad = \quad \bigvee_{i\in\{1..n\}} \mathsf{orch}(\mathsf{T}_i)$$

where, for $i = 1..n$, $\mathsf{T}_i \in \mathsf{rts}(\rho_i \parallel \sigma_i)$ with $\rho \parallel \sigma \xrightarrow{Y:\overline{a}_i}_o \rho' \parallel \sigma'$ and $Y \in \{\mathsf{A}, \mathsf{B}\}$.

$$\mathsf{orch}\Big(\begin{array}{c}\tilde{\rho} \parallel \tilde{\sigma}\\ |\\ \mathsf{T}\end{array}\Big) \quad = \quad \langle \overline{a}, a\rangle.\mathsf{orch}(\mathsf{T})$$

where $\mathsf{T} \in \mathsf{rts}(\rho' \parallel \sigma')$ with $\rho \parallel \sigma \xrightarrow{\mathsf{A}:a}_o \rho' \parallel \sigma'$.

$$\mathsf{orch}\Big(\begin{array}{c}\tilde{\rho} \parallel \tilde{\sigma}\\ |\\ \mathsf{T}\end{array}\Big) \quad = \quad \langle a, \overline{a}\rangle.\mathsf{orch}(\mathsf{T})$$

where $\mathsf{T} \in \mathsf{rts}(\rho' \parallel \sigma')$ with $\rho \parallel \sigma \xrightarrow{\mathsf{B}:a}_o \rho' \parallel \sigma'$.

$$\mathsf{orch}\Big(\begin{array}{c}\tilde{\rho} \parallel \tilde{\sigma}\\ |\\ \mathsf{T}\end{array}\Big) \quad = \quad \langle \alpha, \overline{\alpha}\rangle^+.\mathsf{orch}(\mathsf{T})$$

where $\tilde{\rho} \parallel \tilde{\sigma} = \overline{\alpha}.\rho + \rho' \parallel \alpha.\sigma + \sigma'$ and $\mathsf{T} \in \mathsf{rts}(\rho \parallel \sigma)$.

$$\mathsf{orch}\Big(\begin{array}{c}\tilde{\rho} \parallel \tilde{\sigma}\\ |\\ \boldsymbol{\times}\end{array}\Big) \quad = \quad \boldsymbol{\times}$$

By the above definition it is immediate to check the following fact.

Fact C.15. Let $\mathsf{T} \in \mathsf{rts}(\rho \parallel \sigma)$ such that all its leaves are of the form $\mathbf{0} \parallel \tilde{\sigma}$.
Then $\mathsf{orch}(\mathsf{T})$ is a proper orchestrator, i.e. it does not contain any leaf of the form '$\boldsymbol{\times}$'.

Lemma C.16. *Let* $\mathsf{T} \in \mathsf{rts}(\rho \parallel \sigma)$. *The definition of* $\mathsf{orch}(\mathsf{T})$ *is well-founded*

Proof. By the regularity of T. □

Lemma C.17. *Let* $\mathsf{T} \in \mathsf{rts}(\rho \parallel \sigma)$ *and let* $f = \mathsf{orch}(\mathsf{T})$. *Then* $\mathsf{rt}(\rho \parallel_f \sigma)^- = \mathsf{T}$.

Proof. By construction. □

By Lemma B.3 we have that $\mathsf{rt}_\Sigma(\rho \| \sigma)$ is such that all its leaves are of the form $\mathbf{0} \| \tilde{\sigma}$. Let now $f = \mathsf{orch}(\mathsf{T})$. By Fact C.15, f is a proper orchestrator. By Lemma C.17 we have that $\mathsf{rt}(\rho \|_f \sigma)^- = \mathsf{T}$. So $\mathsf{rt}(\rho \|_f \sigma)^-$, and hence $\mathsf{rt}(\rho \|_f \sigma)$, is such that all its leaves are of the form $\mathbf{0} \| \tilde{\sigma}$. We now get the thesis by Lemma C.7.

C.4. **Proof of Theorem 5.11(4).**

Definition C.18. We define the procedure **O2D** by

$$\mathbf{O2D}(f, \rho, \sigma) \triangleq \mathbf{O2D\text{-}aux}(f, \emptyset, \rho \multimap \sigma)$$

where **O2D-aux** is defined as in Figure 9.

Lemma C.19. *Let f be such that $f : \rho \dashv^{\mathsf{Orch}} \sigma$ holds.*
Then $\boldsymbol{O2D}(f, \rho, \sigma)$ terminates and $\boldsymbol{O2D}(f, \rho, \sigma) :: \rhd \rho \multimap \sigma$

Proof. Similar to the proof of Lemma 5.7. □

O2D-aux$(f, \Gamma, \rho \multimap \sigma)$

if $\rho = \mathbf{1}$ **then** $(\textsc{Ax}) : \dfrac{}{\Gamma \rhd \mathbf{1} \multimap \sigma}$

else **if** $\rho \multimap \sigma \in \Gamma$ **then** $(\textsc{Hyp}) : \dfrac{}{\Gamma, \rho \multimap \sigma \rhd \rho \multimap \sigma}$

else **if** $f = \langle \overline{\alpha}_k, \alpha_k \rangle.f'$ **and** $\rho = \sum_{i \in I} \alpha_i.\rho_i$ **and** $\sigma = \sum_{j \in J} \overline{\alpha}_j.\sigma_j$
and $k \in I \cap J$ **and** $\mathcal{D} = \mathbf{O2D\text{-}aux}(f', \Gamma, \rho \multimap \sigma \rhd \rho_k \multimap \sigma_k) \neq \mathbf{fail}$
then $(+ \cdot +) : \dfrac{\mathcal{D}}{\Gamma \rhd \rho \multimap \sigma}$ **else fail**

else **if** $f = \bigvee_{k \in K} \langle a_k, \overline{a}_k \rangle.f_k$ **and** $\rho = \bigoplus_{i \in I} \overline{a}_i.\rho_i$ **and** $\sigma = \sum_{j \in J} a_j.\sigma_j$
and $K \supseteq I \subseteq J$
and for all $i \in I$ $\mathcal{D}_i = \mathbf{O2D\text{-}aux}(f_i, \Gamma, \rho \multimap \sigma \rhd \rho_i \multimap \sigma_i) \neq \mathbf{fail}$
then $(\oplus \cdot +) : \dfrac{\forall i \in I \ \mathcal{D}_i}{\Gamma \rhd \rho \multimap \sigma}$ **else fail**

else **if** $f = \bigvee_{k \in K} \langle \overline{a}_k, a_k \rangle.f_k$ **and** $\rho = \sum_{i \in I} a_i.\rho_i$ **and** $\sigma = \bigoplus_{j \in J} \overline{a}_j.\sigma_j$
and $I \supseteq J \subseteq K$
and for all $j \in J$ $\mathcal{D}_j = \mathbf{O2D\text{-}aux}(f_j, \Gamma, \rho \multimap \sigma \rhd \rho_j \multimap \sigma_j) \neq \mathbf{fail}$
then $(+ \cdot \oplus) : \dfrac{\forall j \in J \ \mathcal{D}_j}{\Gamma \rhd \rho \multimap \sigma}$ **else fail**

else **fail**

Figure 9: The procedure **O2D-aux**.

Appendix D. Proof of Proposition 5.15 (Correctness and completeness of **Synth**)

Recall that in the algorithm **Synth** (defined in Figure 4) we are not considering orchestrators up to recursion unfolding.

We prove now a version of Proposition 5.15, in which derivability in system $\rhd_{\mathsf{o}}^{\mathsf{x}}$ (defined in Figure 7) is taken into account instead of the orchestrated compliance relation. Given an orchestrator f we denote by $tree(f)$ its corresponding (possibly infinite) regular tree.

Lemma D.1.

(1) *If* $f = \boldsymbol{Synth}(\emptyset, \rho, \sigma) \neq \boldsymbol{fail}$ *then* $\rhd_{\mathsf{o}}^{\mathsf{x}} f : \rho \multimap^{\mathsf{Orch}} \sigma$.
(2) *If* $\rhd_{\mathsf{o}}^{\mathsf{x}} f : \rho \multimap^{\mathsf{Orch}} \sigma$ *then there exists* g *such that* $g = \boldsymbol{Synth}(\emptyset, \rho, \sigma)$ *with* $tree(f) = tree(g)$.

Proof.

(1) Immediate, since the procedure **Synth** is the formalisation of a proof search in System $\rhd_{\mathsf{o}}^{\mathsf{x}}$, as defined in Definition 5.1
(2) Given a derivation tree for $\rhd_{\mathsf{o}}^{\mathsf{x}} f : \rho \multimap^{\mathsf{Orch}} \sigma$, let us consider one path p starting from the root, and such that
   - p ends with an occurrence of (Hyp): $\Gamma', x : \rho' \multimap^{\mathsf{Orch}} \sigma' \rhd_{\mathsf{o}}^{\mathsf{x}} x : \rho' \multimap^{\mathsf{Orch}} \sigma'$
   - p contains more than one other judgments of the form $\Gamma'' \rhd_{\mathsf{o}}^{\mathsf{x}} f'' : \rho' \multimap^{\mathsf{Orch}} \sigma'$.

   If no such a path exists, then any rule in the derivation does precisely correspond to a clause of the algorithm **Synth** and hence the algorithm returns f. Otherwise, since the rules of System $\rhd_{\mathsf{o}}^{\mathsf{x}}$ are such that the proof search is deterministic it is possible to modify the derivation such that in the path from the root to the last judgment of p there is just one other judgment of the form $\Gamma'' \rhd_{\mathsf{o}}^{\mathsf{x}} f'' : \rho' \multimap^{\mathsf{Orch}} \sigma'$. The conclusion of the new derivation will now be $\rhd_{\mathsf{o}}^{\mathsf{x}} g' : \rho \multimap^{\mathsf{Orch}} \sigma$ with $tree(f) = tree(g')$. We can now keep on applying such a procedure until paths like p above no longer exists. □

Now we can get Proposition 5.15 as a corollary of Lemma D.1, using Proposition 5.14, Theorem B.9 and Proposition B.11.

Appendix E. Proof of Theorem 6.22 (Derivations as orchestrator functors)

Form now on, we consider orchestrators, contracts and functors as the (possibly infinite) regular trees they represent. We consider now infinitary versions of $\rhd_{\mathsf{o}}$, $\blacktriangleright$ and $\rhd$.

Definition E.1. We define

- $\rhd_{\mathsf{o}}^{\infty}$ as the system $\rhd_{\mathsf{o}}$ without rule (Hyp);
- $\blacktriangleright^{\infty}$ as the system $\blacktriangleright$ without rule (Hyp-$\ll$);
- $\rhd^{\infty}$ as the system $\rhd$ without rule (Hyp).

Moreover, infinite derivations are allowed in the above systems.

It is not difficult now to check the following lemma.

Lemma E.2.

(1) $\rhd_{\mathsf{o}} f : \rho \multimap^{\mathsf{Orch}} \sigma \;\Leftrightarrow\; \rhd_{\mathsf{o}}^{\infty} f : \rho \multimap^{\mathsf{Orch}} \sigma$
(2) $\blacktriangleright\ \sigma \ll \sigma' \;\Leftrightarrow\; \blacktriangleright^{\infty}\ \sigma \ll \sigma'$

We now prove Theorem 6.22 by proceeding as follows: We first define a proof reconstruction procedure $\mathbf{R}^{\infty}$ taking as argument two derivations $\mathcal{D}' :: \rhd_{\mathsf{o}}^{\infty} f : \rho \rightharpoonup^{\mathsf{Orch}} \sigma$ and $\mathcal{D}'' :: \blacktriangleright^{\infty} \sigma \ll \sigma'$, and, in case it does not fails, it produces a (possibly infinite) derivation $\mathcal{D}'''$ in system $\rhd^{\infty}$ partially decorated with orchestration actions. We then show that $\mathbf{R}$ does not fail and that the derivation $\mathcal{D}'''$ can be easily turned in a derivation $\widetilde{\mathcal{D}'''}$ is such that $\widetilde{\mathcal{D}'''} :: \rhd_{\mathsf{o}}^{\infty} \mathbf{F}(f) : \rho \rightharpoonup^{\mathsf{Orch}} \sigma$.

Theorem 6.22 hence descends immediately by Theorem B.9 and Lemma E.2.

Definition E.3 (The algorithm $\mathbf{R}^{\infty}$)**.** Let $\mathcal{D}' :: \rhd_{\mathsf{o}}^{\infty} f : \rho \rightharpoonup^{\mathsf{Orch}} \sigma$ and $\mathcal{D}'' :: \blacktriangleright^{\infty} \sigma \ll \sigma'$. The algorithm $\mathbf{R}$ is defined by

$$\mathbf{R}^{\infty}(\mathcal{D}', \mathcal{D}'') \;=\; \mathbf{R}^{\infty}\textbf{-aux}(\mathcal{D}', \mathcal{D}'', \emptyset)$$

where $\mathbf{R}^{\infty}$**-aux** is a procedure with an extra argument (an environment), defined by cases according to the following clauses.We name the clauses with the name of the last rules applied in the first two derivations.The algorithm fails in case no clause can be applied.

Clause $*$**-**(Ax **-**$\ll$)**:**

$$\mathbf{R}^{\infty}\textbf{-aux}\Big(\dfrac{\boxed{\mathcal{D}_1}}{\Gamma_1 \rhd_{\mathsf{o}}^{\mathsf{x}} f : \rho \rightharpoonup^{\mathsf{Orch}} \sigma},\ \dfrac{}{\Gamma_2 \blacktriangleright \mathbf{1} \ll \sigma'}\,(\textsc{Ax}),\ \Gamma_3\Big) = \dfrac{}{\Gamma_3 \rhd^{\infty} \mathbf{1} : \mathbf{1} \rightharpoonup \sigma'}\,(\textsc{Ax})$$

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 $(+ \cdot \oplus)$**-**$(\oplus \cdot + $**-**$\ll)$**:**

$$\mathbf{R}^{\infty}\textbf{-aux}\Big(\ \dfrac{\boxed{\mathcal{D}'_i}}{\Gamma'_1 \rhd_{\mathsf{o}}^{\infty} f_i : \rho_i \rightharpoonup^{\mathsf{Orch}} \sigma_i\ (\forall i \in I)}\,(+\cdot\oplus)\ ,$$
$$\dfrac{\Gamma'_1 \rhd_{\mathsf{o}}^{\infty} f_i : \rho_i \rightharpoonup^{\mathsf{Orch}} \sigma_i\ (\forall i \in I)}{\Gamma_1 \rhd_{\mathsf{o}}^{\infty} \bigvee_{i\in I}\langle \overline{a}_i, a_i\rangle.f_i : \Sigma_{j\in I\cup J} a_j.\rho_j \rightharpoonup^{\mathsf{Orch}} \bigoplus_{i\in I} \overline{a}_i.\sigma_i}\,(+\cdot\oplus)$$
$$\text{where } \Gamma'_1 = \Gamma_1, \bigvee_{i\in I}\langle \overline{a}_i, a_i\rangle.f_i : \textstyle\sum_{j\in I\cup J} a_j.\rho_j \rightharpoonup^{\mathsf{Orch}} \oplus_{i\in I} \overline{a}_i.\sigma_i$$

$$\dfrac{(h \in I \cap K)\quad \boxed{\mathcal{D}''_h}\quad \Gamma'_2 \blacktriangleright^{\infty} \sigma_h \ll \sigma'_h}{\Gamma_2 \blacktriangleright^{\infty} \bigoplus_{i\in I} \overline{a}_i.\sigma_i \ll \Sigma_{k\in K} \overline{a}_k.\sigma'_k}\,(\oplus\cdot+\text{-}\ll)\ ,\Gamma_3\ \Big) =$$
$$\text{where } \Gamma'_2 = \Gamma_2, \textstyle\bigoplus_{i\in I} \overline{a}_i.\sigma_i \ll \Sigma_{k\in K}\overline{a}_k.\sigma'_k$$

$$\dfrac{\boxed{\mathcal{D}_h}}{\Gamma_3, \rho \rightharpoonup \sigma' \rhd^{\infty} \mu : \rho_h \rightharpoonup^{\mathsf{Orch}} \sigma'_h}\quad (h \in I \cap K)\,(+\cdot+)$$
$$\Gamma_3 \rhd^{\infty} \langle \overline{a}_h, a_h\rangle^{+} : \Sigma_{j\in I\cup J} a_j.\rho_j \rightharpoonup^{\mathsf{Orch}} \Sigma_{k\in K}\overline{a}_k.\sigma'_k$$

where

$$\dfrac{\boxed{\mathcal{D}_h}}{\Gamma_3, \rho \rightharpoonup \sigma' \rhd^{\infty} \mu : \rho_h \rightharpoonup^{\mathsf{Orch}} \sigma'_h} = \mathbf{R}^{\infty}\textbf{-aux}\Big(\boxed{\mathcal{D}'_i}\ \dfrac{}{\Gamma'_1 \rhd_{\mathsf{o}}^{\infty} f_i : \rho_i \rightharpoonup^{\mathsf{Orch}} \sigma_i},\ \boxed{\mathcal{D}''_h}\ \dfrac{}{\Gamma'_2 \blacktriangleright \sigma_h \ll \sigma'_h},\ (\Gamma_3, \rho \rightharpoonup \sigma')\Big)$$

and $\rho = \Sigma_{j\in I\cup J} a_j.\rho_j$ and $\sigma' = \Sigma_{k\in K} a_k.\sigma'_k$.

Clause $(+ \cdot +)$**-**$(+ \cdot +$**-**$\ll)$**:**

$$\mathbf{R}^{\infty}\text{-}\mathbf{aux}\Bigg(\quad \dfrac{\boxed{\mathcal{D}_p}}{\Gamma_1' \rhd_{\mathsf{o}}^{\infty} f' : \rho_p \multimap^{\mathsf{Orch}} \sigma_p \qquad (p \in I \cap K)}\,(+\cdot+) \;,$$
$$\Gamma_1 \rhd_{\mathsf{o}}^{\infty} \langle \alpha_p, \overline{\alpha}_p \rangle^{+}.f' : \Sigma_{k\in K}\overline{\alpha}_k.\rho_k \multimap^{\mathsf{Orch}} \Sigma_{i\in I}\alpha_i.\sigma_i$$
$$\text{where } \Gamma_1' = \Gamma_1, \langle \alpha_p, \overline{\alpha}_p \rangle^{+}.f' : \Sigma_{k\in K}\overline{\alpha}_k.\rho_k \multimap^{\mathsf{Orch}} \Sigma_{i\in I}\alpha_i.\sigma_i$$

$$\dfrac{\boxed{\tilde{\mathcal{D}}_i}}{\Gamma_2' \;\blacktriangleright^{\infty}\; \sigma_i \ll \sigma_i' \qquad (\forall i \in I)}\,(+\cdot+\text{-}\preccurlyeq) \;,\; \Gamma_3 \;\Bigg) =$$
$$\Gamma_2 \;\blacktriangleright^{\infty}\; \Sigma_{i\in I}\alpha_i.\sigma_i \ll \Sigma_{j\in I\cup J}\alpha_j.\sigma_j'$$
$$\text{where } \Gamma_2' = \Gamma_2, \Sigma_{i\in I}\alpha_i.\sigma_i \ll \Sigma_{j\in I\cup J}\alpha_j.\sigma_j'$$

$$\dfrac{\boxed{\mathcal{D}_p'}}{\Gamma_3, \rho \multimap \sigma' \rhd^{\infty} \mu : \rho_p \multimap \sigma_p' \qquad (p \in (I \cup J) \cap K)}\,(+\cdot+)$$
$$\Gamma_3 \rhd^{\infty} \langle \alpha_p, \overline{\alpha}_p \rangle^{+} : \rho \multimap \sigma'$$

where

$$\boxed{\mathcal{D}_p'} \;\; \Gamma_3, \rho \multimap^{\mathsf{Orch}} \sigma' \rhd^{\infty} \mu : \rho_p \multimap \sigma_p' \;=\; \mathbf{R}^{\infty}\text{-}\mathbf{aux}\big(\boxed{\mathcal{D}_p} \;\; \Gamma_1' \rhd_{\mathsf{o}}^{\infty} f' : \rho_p \multimap^{\mathsf{Orch}} \sigma_p,\; \boxed{\tilde{\mathcal{D}}_p} \;\; \Gamma_2' \;\blacktriangleright^{\infty}\; \sigma_p \ll \sigma_p',\; (\Gamma_3, \rho \multimap \sigma')\big)$$

and $\rho = \Sigma_{k\in K}\overline{\alpha}_k.\rho_k$ and $\sigma' = \Sigma_{j\in I\cup J}\alpha_j.\sigma_j'$,

Clause $(\oplus\cdot+)$-$(+\cdot+$-$\ll)$:
The construction follows a definition pattern similar to those of the previous clauses.

Clause $(+\cdot\oplus)$-$(\oplus\cdot\oplus$-$\ll)$:
The construction follows a definition pattern similar to those of the previous clauses.

Proposition E.4.
Let $\mathcal{D}' :: \rhd_{\mathsf{o}}^{\infty} f : \rho \multimap^{\mathsf{Orch}} \sigma$ *and* $\mathcal{D}'' :: \blacktriangleright^{\infty} \sigma \ll \sigma'$, *and let* $\mathbf{F}_{\mathcal{D}''}$ *be defined as in Definition 6.18.*

(1) *The computation of* $\mathbf{R}^{\infty}(\mathcal{D}', \mathcal{D}'')$ *never fails.*
(2) $\mathbf{R}^{\infty}(\mathcal{D}', \mathcal{D}'') = \mathcal{D} :: \rhd^{\infty} \rho \multimap \sigma'$, *where* $\mathcal{D}$ *is decorated with orchestration actions. Moreover, out of* $\mathcal{D}$ *it is possible to get* $\widetilde{\mathcal{D}}$ *such that* $\widetilde{\mathcal{D}} :: \rhd^{\infty} \mathbf{F}_{\mathcal{D}''}(f) : \rho \multimap^{\mathsf{Orch}} \sigma'$.

Proof. By inspection of the clauses of the procedure, the computation never fails if we start from $\mathcal{D}'$ and $\mathcal{D}''$ as above.
Out of the (possible infinite) decorated derivation $\mathcal{D}$ is is possible to get an orchestrator $\tilde{f}$ such that $\tilde{f} : \rho \multimap^{\mathsf{Orch}} \sigma'$ (because of the regularity of the derivation tree $\mathcal{D}$) and a derivation $\widetilde{\mathcal{D}} :: \rhd^{\infty} \tilde{f} : \rho \multimap^{\mathsf{Orch}} \sigma'$. Besides, working on the form of the clauses of the procedure, it can be shown that $\tilde{f} = \mathbf{F}_{\mathcal{D}''}(f)$. □

Theorem 6.22 descends now as a corollary from the above proposition.