A theory of retractable and speculative contracts

Franco Barbanera, Ivan Lanese, Ugo De'Liguoro

A Theory of Retractable and Speculative Contracts

Franco Barbanera[a], Ivan Lanese[b,*], Ugo de'Liguoro[c]

[a] *Dipartimento di Matematica e Informatica, University of Catania.*
[b] *Dipartimento di Informatica - Scienza e Ingegneria, University of Bologna/INRIA*
[c] *Dipartimento di Informatica, University of Torino.*

Abstract

Behavioral contracts are abstract descriptions of expected communication patterns followed by either clients or servers during their interaction. Behavioral contracts come naturally equipped with a notion of *compliance*: when a client and a server follow compliant contracts, their interaction is guaranteed to progress or successfully complete. We study two extensions of behavioral contracts, *retractable contracts* dealing with *backtracking* and *speculative contracts* dealing with *speculative execution*. We show that the two extensions give rise to *the same notion of compliance*. As a consequence, they also give rise to the same *subcontract relation*, which determines when one server can be replaced by another preserving compliance. Moreover, compliance and subcontract relation are both decidable in quadratic time. Finally, we study the relationship between retractable contracts and calculi for reversible computing.

Keywords: Behavioral contracts, backtracking, speculative execution, compliance, reversible computing

1. Introduction

Binary behavioral contracts [1, 2, 3] and binary session types [4] are abstractions of programs used to statically ensure that a client and a server successfully interact (see the survey in [5]). Along the years, the basic theory has been extended to deal with many features of clients and servers, such as exceptions [6], time [7], and so on. We consider here two new features: *backtracking*, allowing one to go back to previous stages of the interaction, and *speculative execution* [8], allowing one to try different alternatives concurrently. These two features have quite different origin and aims. Backtracking is used to avoid failures due to wrong past decisions in a wide range of settings, from the undo button in web browsers, to the execution model of Prolog, to techniques for rollback-

*Corresponding author
Email addresses: barba@di.unica.it (Franco Barbanera), ivan.lanese@gmail.com (Ivan Lanese), ugo.deliguoro@unito.it (Ugo de'Liguoro)

recovery [9]. Speculative execution is used for efficiency reasons in different areas, from simulation [10], to thread-level optimization [11], to web services [12].

We present two extensions of binary contracts (Section 2): *retractable contracts* capturing backtracking, and *speculative contracts* capturing speculative execution. The two extensions are based on the same syntax, but naturally have different semantics. Essentially, they add to the session contracts of [13, 14] (called first-order session behaviors in [13]) an operator of *external choice among output* operations. Classically, external means that the participant provides a set of alternatives, and the communication partner decides which one (s)he wants to take. This is opposed to internal choice, where the participant decides in isolation which alternative is taken. The setting in our extension is slightly more complex. The most interesting case is when an external choice among outputs and an external choice among inputs interact. In the retractable semantics, the client and the server agree on which option to explore, but they rollback and try a different possibility if the computation gets stuck. In the speculative semantics, instead, all the possibilities are explored concurrently, and it is enough for one of them to succeed in order to guarantee the success of the whole computation.

This paper defines retractable and speculative contracts, and studies the related theory, considering the notions of *compliance* (Section 3), guaranteeing that the interaction progresses or successfully completes, *subcontract relation* (Section 4), determining when a server (resp. client) can be replaced by another server (resp. client) preserving compliance, and *dual contract* (Section 4), that is the most general contract (in terms of the subcontract relation) compliant with a given contract. Our analysis provides two main insights:

- Even if retractable contracts and speculative contracts have different semantics and give rise to different client-server interactions, the relations of compliance, subcontract and duality in the two settings do coincide. While surprising at first sight, this can be explained by noticing that in both the cases different alternatives are explored (sequentially for retractable contracts, in parallel for speculative contracts) and the success of one of them guarantees the success of the whole computation. In other terms, the two semantics provide different implementations of *angelic nondeterminism*, first described by Hoare [15].

- While retractable/speculative contracts are strictly more expressive than session contracts (indeed they are a conservative extension, see Section 3.1), their theory preserves the main good properties of the theory of session contracts. In particular, compliance and subcontract relations are both decidable (Section 3) in quadratic time (Section 5), and the dual of a contract always exists and has a simple syntactic characterization (Section 4).

A natural way to ensure the existence of the dual contract is to introduce an operator of internal choice among inputs. While this operator has limited practical impact, it makes the model more symmetric and the mathematical treatment simpler.

The results above make us confident in the fact that our semantics correctly captures the interaction patterns we are interested in. As further element supporting this, we show (Section 6) that the backtracking mechanism of retractable contracts can be seen as an application to behavioral contracts of the general theory proposed in [16] to define reversible extensions of process calculi.

This paper is an extended and revised version of [17] (a few preliminary results had been originally presented in a workshop paper [18]). Section 6, where the relation between retractable contracts and calculi for reversible computing is investigated, is completely new. Moreover, the analysis of the complexity of deciding compliance and subcontract relation has been refined, reducing the resulting complexity from a fifth power to a square. The paper also includes additional proofs and examples, and a more detailed discussion of related work. Finally, the whole presentation has been revised and improved.

Proofs omitted from the main part are collected in Appendix A.

2. Contracts for Retractable and Speculative Interactions

We present below a uniform syntax for retractable and speculative contracts, with two semantics. It can be obtained from the syntax of (first-order) session contracts of [13, 14] – dubbed SC in the present paper, and briefly recalled in Section 3.1.1 – by just adding external retractable/speculative choice among outputs and internal choice among inputs. As a matter of fact our contracts can also be seen as an extension of the retractable session contracts of [18], that we dub here $\mathbf{rC}$, by simply adding internal choice among inputs.

Definition 1 (Retractable/Speculative Contracts). Let $\mathcal{N}$ *(set of names)* be some countable set of symbols and let $\overline{\mathcal{N}}$ *(set of conames)* be $\{\overline{a} \mid a \in \mathcal{N}\}$, with $\mathcal{N} \cap \overline{\mathcal{N}} = \emptyset$. The set $\mathbf{rs}\mathsf{C}$ of *retractable/speculative contracts* is defined as the set of the *closed* expressions generated by the following grammar,

$$
\begin{array}{rcll}
\sigma, \rho & := & \mid \mathbf{1} & \text{SUCCESS} \\
& & \mid \sum_{i \in I} a_i.\sigma_i & \text{EXTERNAL INPUT CHOICE} \\
& & \mid \sum_{i \in I} \overline{a}_i.\sigma_i & \text{EXTERNAL OUTPUT CHOICE} \\
& & \mid \bigoplus_{i \in I} a_i.\sigma_i & \text{INTERNAL INPUT CHOICE} \\
& & \mid \bigoplus_{i \in I} \overline{a}_i.\sigma_i & \text{INTERNAL OUTPUT CHOICE} \\
& & \mid x & \text{VARIABLE} \\
& & \mid \mathsf{rec}\, x.\sigma & \text{RECURSION}
\end{array}
$$

where I is non-empty and finite, the names and the conames in choices are pairwise distinct and σ is not a variable in $\mathsf{rec}\, x.\sigma$. Recursion in $\mathbf{rs}\mathsf{C}$ is guarded and hence contractive in the usual sense. We take an equi-recursive view of recursion by equating $\mathsf{rec}\, x.\sigma$ with $\sigma[\mathsf{rec}\, x.\sigma/x]$.

Intuitively, a name a represents a communication channel, which can be used either in an input action (denoted a) or in an output action (denoted $\overline{a}$). The dot is used to denote precedence: to perform $a.\sigma$ one first performs a and then

continues as specified by σ. In the syntax above, branches $a_i.\sigma_i$ and $\overline{a}_i.\sigma_i$ can be composed either in internal choice or in external choice. In internal choice the participant non-deterministically decides which branch he wants to take. In external input choice the participant presents a set of options and waits for his communication partner to decide which one to take. External output choice is the main novelty of this paper, and its behavior depends on whether the retractable or the speculative semantics is chosen. Hence, we will describe it in detail in the presentation of the two semantics.

We use α to range over $\mathcal{N} \cup \overline{\mathcal{N}}$, with the convention $\overline{\alpha} = \overline{a}$ if $\alpha = a$, and $\overline{\alpha} = a$ if $\alpha = \overline{a}$. We write $\alpha_1.\sigma_1 + \alpha_2.\sigma_2$ for binary external input/output choice and $\alpha_1.\sigma_1 \oplus \alpha_2.\sigma_2$ for binary internal input/output choice. They are both commutative by definition. Also, $\alpha.\sigma$ denotes both internal and external unary choice. This entails no ambiguity, since internal and external choices do coincide in the unary case. We also write $\alpha_k.\sigma_k + \sigma'$ for $\sum_{i\in I} \alpha_i.\sigma_i$ where $k \in I$ and $\sigma' = \sum_{i\in(I\setminus\{k\})} \alpha_i.\sigma_i$ (and similarly for internal choices). When no ambiguity can arise, we call just *contracts* the expressions in $\mathbf{rsC}$. They are written by omitting all trailing $\mathbf{1}$'s.

We discuss below the two interpretations and the two semantics for our contracts: the retractable one, and the speculative one.

2.1. Retractable semantics

The main novelty of the retractable semantics is that, when an external choice among outputs and an external choice among inputs interact, the client and the server agree on which option to explore *first*, but they rollback and try a different possibility if the computation gets stuck.

In order to deal with rollbacks, we decorate contracts with their *history*. A history H represents the alternatives that have been discharged in previous retractable choices (hence the branches that can be tried upon rollbacks due to synchronization failures). Histories are formalized as stacks, since we always rollback to the last retractable choice made. A current contract σ with history H is represented by $\mathrm{H} \ltimes \sigma$. We use the symbol '$\circ$' to stand for "no-remaining-alternatives".

Definition 2 (Contracts with History). Let $\mathsf{Histories}$ be the expressions generated by the grammar $\mathrm{H} ::= \langle\,\rangle \mid \mathrm{H} : \sigma$, where $\sigma \in \mathbf{rsC} \cup \{\circ\}$ and $\circ \notin \mathbf{rsC}$. $\mathsf{Histories}$ are hence stacks of contracts and $\circ$. Then the set of *contracts with history* is defined by:

$$\mathbf{rsCH} = \{\, \mathrm{H} \ltimes \sigma \mid \mathrm{H} \in \mathsf{Histories}, \sigma \in \mathbf{rsC} \cup \{\circ\} \,\}$$

We write just $\sigma_1 : \cdots : \sigma_k$ for the stack $(\cdots(\langle\,\rangle : \sigma_1) : \cdots) : \sigma_k$. Moreover, we simply write σ for $\langle\,\rangle : \sigma$ and $\mathrm{H}_1 : \mathrm{H}_2$ for $(\cdots(\mathrm{H}_1 : \sigma_1) : \cdots) : \sigma_k$ where $\mathrm{H}_2 = \sigma_1 : \cdots : \sigma_k$.

As standard for contracts, the definition of the retractable semantics is in two stages: we first define a labeled transition system (LTS) for contracts with history (Definition 3), and then we build on top of it a reduction semantics for pairs of contracts modeling one client and one server (Definition 4).

Definition 3 (Semantics of Contracts with History).

$$(+)\quad \mathrm{H}\ltimes \alpha.\sigma + \sigma' \xrightarrow{\alpha} \mathrm{H}:\sigma'\ltimes\sigma \qquad (\oplus)\quad \mathrm{H}\ltimes \alpha.\sigma \oplus \sigma' \xrightarrow{\tau} \mathrm{H}\ltimes\alpha.\sigma$$

$$(\alpha)\quad \mathrm{H}\ltimes \alpha.\sigma \xrightarrow{\alpha} \mathrm{H}:\circ\ltimes\sigma \qquad (\mathsf{rb})\quad \mathrm{H}:\sigma'\ltimes\sigma \xrightarrow{\mathsf{rb}} \mathrm{H}\ltimes\sigma'$$

In the transition rule for external choice $(+)$, the action α is executed, and the discharged branches in σ' are memorized on top of the history H. In internal choice $(\oplus)$, instead, the selection of one branch is represented by a label τ, and the history H is unchanged. When a unary choice is executed (α), a '$\circ$' is added to the history, meaning that the only possible branch has been tried and no alternative is left. Rule (rb) pops the contract at the top of the stack, replacing the current one with it.

The client/server interaction is modeled by the reduction of their parallel composition, that can be either *forward*, consisting of CCS-style synchronizations and single internal choices, or *backward*, only when there is no possible forward reduction, and the client is not satisfied, i.e., it is different from **1**.

Definition 4 (Semantics of Retractable Client/Server Pairs).
The following rules, plus the rule symmetric to (τ) w.r.t. $\|$, define the relation $\longrightarrow$ over pairs of contracts with history:

$$(\mathit{comm})\ \frac{\mathrm{H}_1\ltimes\rho \xrightarrow{\alpha} \mathrm{H}_1'\ltimes\rho' \qquad \mathrm{H}_2\ltimes\sigma \xrightarrow{\overline{\alpha}} \mathrm{H}_2'\ltimes\sigma'}{\mathrm{H}_1\ltimes\rho \parallel \mathrm{H}_2\ltimes\sigma \longrightarrow \mathrm{H}_1'\ltimes\rho' \parallel \mathrm{H}_2'\ltimes\sigma'} \qquad (\tau)\ \frac{\mathrm{H}_1\ltimes\rho \xrightarrow{\tau} \mathrm{H}_1\ltimes\rho'}{\mathrm{H}_1\ltimes\rho \parallel \mathrm{H}_2\ltimes\sigma \longrightarrow \mathrm{H}_1\ltimes\rho' \parallel \mathrm{H}_2\ltimes\sigma}$$

$$(\mathit{rbk})\ \frac{\mathrm{H}_1\ltimes\rho \xrightarrow{\mathsf{rb}} \mathrm{H}_1'\ltimes\rho' \qquad \mathrm{H}_2\ltimes\sigma \xrightarrow{\mathsf{rb}} \mathrm{H}_2'\ltimes\sigma' \qquad \rho \neq \mathbf{1}}{\mathrm{H}_1\ltimes\rho \parallel \mathrm{H}_2\ltimes\sigma \longrightarrow \mathrm{H}_1'\ltimes\rho' \parallel \mathrm{H}_2'\ltimes\sigma'}$$

Rule (rbk) applies only if neither (comm) nor (τ) do.

The *forward reduction* $\longrightarrow_f$ is the relation generated by rules (τ) and (comm). The *backward reduction* $\longrightarrow_b$ is the relation generated by rule (rbk).

Remark 1. The relation $\longrightarrow$ is defined by means of a system of rules with priorities. Reduction relations with priority rules have been initially introduced in the context of term-rewriting [19] and of languages with pattern-matching [20]. Orderings on SOS rules were later proposed by Phillips and Ulidowski as an alternative to negative premises [21, 22]. In fact alternative definitions would either consist in resorting to a "negated" reduction to specify that rollback can be applied when the forward semantics is stuck; or in introducing error states reached when a client and a server exhibit incompatible choices.

We opted for the present definition in order to get a simpler treatment of the reduction relation and hence simpler and more readable proofs.

Example 1. In order to get a better insight into the role of $\circ$ in the rollback mechanism, observe that, for a client like $a.c + b.d$, rule $(+)$ in Definition 3 forces the memorization of the discharged branch, say $a.c$, independently from the

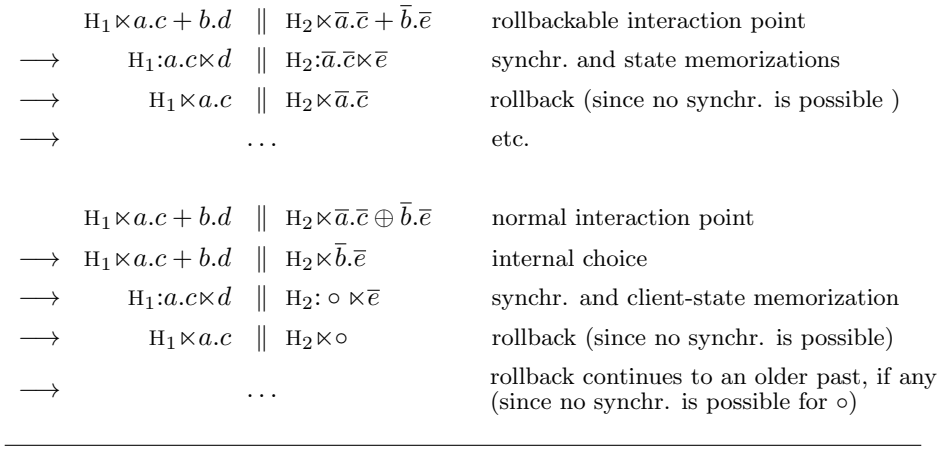

Figure 1: Rollback

shape of the server. If the server is, for instance, $\overline{a}.\overline{c} + \overline{b}.\overline{e}$ (see the first reduction sequence in Figure 1) then the server memorizes a discarded branch too, $\overline{a}.\overline{c}$ in the example. Hence, after the synchronization failure of d and $\overline{e}$, the two discarded branches are recovered and executed. If the server is, instead, $\overline{a}.\overline{c} \oplus \overline{b}.\overline{e}$ (see the second reduction sequence in Figure 1) then the server is not willing to rollback this choice, and it stores a $\circ$ in the history. Upon synchronization failure, the discarded branch of the client and the $\circ$ of the server are recovered, but, since $\circ$ cannot reduce, this immediately causes a new synchronization failure and an *older* past (if any) is recovered. Summarizing, recovered branches are actually executed only if both the client and the server are willing to do so.

Remark 2. The semantics defined above for retractable client/server pairs can be seen as an instantiation on contracts of the standard reversible semantics for process calculi, see, e.g., [23, 16, 24, 25]. In particular, the semantics would become a classic uncontrolled semantics (according to the terminology in [25]) by removing the four control mechanisms below:

1. the fact that only external choices are retractable;
2. the side condition $\rho \neq \mathbf{1}$ in rule (*rbk*), which disallows backtrack after success;
3. the fact that rule (*rbk*) can be applied only if no other rule applies, ensuring that backtrack is enabled only when no forward reduction is possible;
4. the fact that in external choices the selected path is not stored in the history, so that each path can be tried at most once.

These mechanisms provide a semantic control of reversibility [25], specifying which rollback steps are allowed, and when. We formalize this intuition in

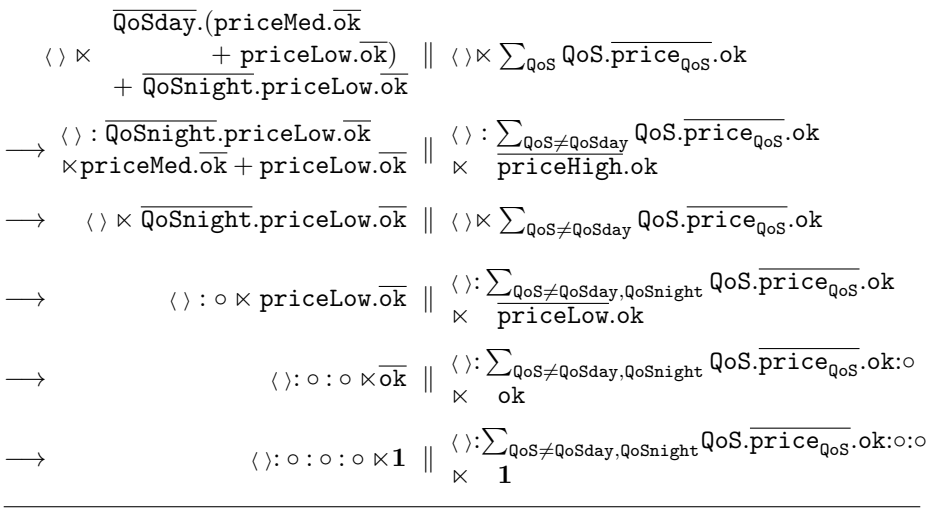

Figure 2: An example of retractable interaction

Section 6. We discuss in Remark 3 the impact that removing the above control mechanisms would have on retractable contracts and on their theory.

Example 2. Retractable contracts allow one to first try a preferred alternative, but to accept also another alternative if the first one proves to be impossible to obtain. In cloud computing settings, companies may hire virtual machines and storing facilities from cloud providers with some agreed Quality of Service (QoS). A company is willing to hire at some medium or low price a certain amount of machines for online elaboration during the daytime, but, if the price is too high, it is also willing to switch to offline night elaboration. In this last case it is only willing to pay a low price.

A retractable contract with this behavior may be written as:

$$\mathsf{cloudClient} = \overline{\mathtt{QoSday}}.(\mathtt{priceMed}.\overline{\mathtt{ok}} + \mathtt{priceLow}.\overline{\mathtt{ok}}) + \overline{\mathtt{QoSnight}}.\mathtt{priceLow}.\overline{\mathtt{ok}}$$

Notice that the contract does not specify which alternative the client prefers: this aspect of the client behavior is abstracted away[1]. A sample server is:

$$\mathsf{cloudServer} = \textstyle\sum_{\mathtt{QoS}\in\{\mathtt{QoSday},\mathtt{QoSnight},\dots\}} \mathtt{QoS}.\overline{\mathtt{price}_{\mathtt{QoS}}}.\mathtt{ok}$$

A sample interaction is described in Figure 2, where we assume that

$$\mathtt{price}_{\mathtt{QoSday}} = \mathtt{priceHigh} \quad \text{and} \quad \mathtt{price}_{\mathtt{QoSnight}} = \mathtt{priceLow}$$

In particular, the interaction starts from a rollbackable interaction point (1) and proceeds as follows: (2) synchronization on $\mathtt{QoSday}$ and memorization of the

[1] As we will discuss in Section 7, it is possible to extend our contracts to express an order of preference among external output choices, as done in [26] in the context of session types.

discarded `QoSnight` branch; (3) rollback, since no choice has been offered for the requested $\overline{\texttt{priceHigh}}$; (4) synchronization on `QoSnight` and memorization of $\circ$ (no further alternative branch is available); (5) synchronization on `priceLow`; (6) synchronization on `ok` and success state reached (client is satisfied).

2.2. Speculative semantics

The main idea of the speculative semantics is that in an external output choice all the options are tried concurrently: if at least one of them succeeds, then the whole computation succeeds. In order to represent concurrent trials we need runtime contracts featuring multiple *threads*. A thread is a contract preceded by zero or more prefixes of the form $\alpha\,@$, meaning that the action α has been performed in the past.

Definition 5 (Contracts with Threads). *Contracts with threads* $\mathbf{C}$, used as runtime syntax for contracts, are parallel compositions of *threads* $\mathbf{T}$. Each thread is a contract prefixed by zero or more actions.

$$\mathbf{C} ::= \mathbf{T} \mid (\mathbf{C} \mid \mathbf{T}) \mid (\mathbf{T} \mid \mathbf{C}) \qquad \mathbf{T} ::= \sigma \mid \alpha\,@\,\mathbf{T}$$

We assume the operator '$\mid$' to be associative and commutative.

We will always work with contracts with threads such that threads are uniquely identified by the sequence of actions prefixing them. Our prefixes may remind location prefixes from [27], however our prefixes are deterministically generated from performed actions, and they do not represent locations.

As for the retractable semantics, the definition of the speculative semantics is in two stages: we first define an LTS for contracts with threads (Definition 6), and then we use it to define a reduction semantics for pairs of contracts with threads representing one client and one server (Definition 7).

Definition 6 (Semantics of Contracts with Threads).
The following rules define the LTS for contracts with threads. In the LTS, we use as labels: $\alpha ::= a \mid \overline{a}$ for actions, $\beta ::= \alpha \mid \alpha\beta$ for sequences of actions, and $\beta_\tau ::= \tau \mid \beta \mid \beta, \mathbf{T}$ for general labels.

$$\text{(Fork)}\ \alpha.\sigma + \sigma' \xrightarrow{\alpha,\sigma'} \alpha\,@\,\sigma \qquad \text{($\oplus$)}\ \alpha.\sigma \oplus \sigma' \xrightarrow{\tau} \alpha.\sigma \qquad \text{(α)}\ \alpha.\sigma \xrightarrow{\alpha} \alpha\,@\,\sigma$$

$$\text{(@-α)}\ \frac{\mathbf{T} \xrightarrow{\beta} \mathbf{T}'}{\alpha\,@\,\mathbf{T} \xrightarrow{\alpha\beta} \alpha\,@\,\mathbf{T}'} \qquad \text{(@-α-T)}\ \frac{\mathbf{T} \xrightarrow{\beta,\mathbf{T}''} \mathbf{T}'}{\alpha\,@\,\mathbf{T} \xrightarrow{\alpha\beta,\alpha\,@\,\mathbf{T}''} \alpha\,@\,\mathbf{T}'}$$

$$\text{(@-τ)}\ \frac{\mathbf{T} \xrightarrow{\tau} \mathbf{T}'}{\alpha\,@\,\mathbf{T} \xrightarrow{\tau} \alpha\,@\,\mathbf{T}'} \qquad \text{(ParL)}\ \frac{\mathbf{T} \xrightarrow{\beta_\tau} \mathbf{T}'}{\mathbf{T} \mid \mathbf{C} \xrightarrow{\beta_\tau} \mathbf{T}' \mid \mathbf{C}}$$

In the rule for external choice (*Fork*), when an action α is executed, its continuation σ is prefixed by it. The other branches σ' need to be executed

in a freshly spawned thread. Since such thread needs to be installed at top level, σ' is added to the label, and the actual installation is performed at the level of speculative client/server pairs (see rule (*comm*) in Definition 7). The rule for internal choice ($\oplus$) simply selects one of the available options. A unary choice (α) executes the action α and prefixes with it the continuation σ. Rules (@-α), (@-α-T), and (@-τ) enable execution below an @ prefix. In particular, in rule (@-α), the prefix itself is added to the label β. Prefixes uniquely identify threads, and ensure that each thread interacts only with the one with dual prefix which is running on the communication partner. This is specified in Definition 7 below. Rule (@-α-T) is analogous to rule (@-α), but the label also contains a thread $\mathbf{T}''$, and the prefix α is added to both β and $\mathbf{T}''$. No prefix is added to τ actions, propagated by rule (@-τ). Rule (ParL) simply allows components of a parallel composition to execute (a symmetric rule is not needed thanks to the commutativity of $|$).

The interaction of a client with a server is modeled by the reduction of their parallel composition.

Definition 7 (Semantics of Speculative Client/Server Pairs).
The following rules, plus the rule symmetric to (τ) w.r.t. $\|$, define the relation $\longrightarrow$ over pairs of contracts with threads. In the LTS below, $?\mathbf{T}$ denotes either the thread $\mathbf{T}$ or nothing. Hence, $\beta, ?\mathbf{T}$ and $\mathbf{C} \,|?\mathbf{T}$ are respectively β and $\mathbf{C}$ if $?\mathbf{T}$ is nothing, and $\beta, \mathbf{T}$ and $\mathbf{C} \mid \mathbf{T}$ otherwise. Also, the duality operator extends from actions to sequences: $\overline{\alpha\beta} = \overline{\alpha}\overline{\beta}$.

$$\text{(comm)}\ \frac{\mathbf{C} \xrightarrow{\beta, ?\mathbf{T}} \mathbf{C}' \quad \mathbf{C}'' \xrightarrow{\overline{\beta}, ?\mathbf{T}''} \mathbf{C}'''}{\mathbf{C} \parallel \mathbf{C}'' \longrightarrow \mathbf{C}' \,|?\mathbf{T} \parallel \mathbf{C}''' \,|?\mathbf{T}''} \qquad (\tau)\ \frac{\mathbf{C} \xrightarrow{\tau} \mathbf{C}'}{\mathbf{C} \parallel \mathbf{C}'' \longrightarrow \mathbf{C}' \parallel \mathbf{C}''}$$

Rule (*comm*) allows threads performing dual sequences of actions to interact. This implies that both the performed actions and the prefixes of the threads executing them have to be dual. Threads in the labels, if present, are installed in parallel. Rule (τ) simply propagates the τ action.

Example 3. A server provides access to multiple algorithms for SAT solving [28]. A client first sends the problem instance to be solved, then selects the algorithm, and finally sends the relevant parameters. The server computes the solution according to the received commands, and sends it back. Since the most efficient technique depends on the problem instance [29], the server supports speculative execution to allow one to try different algorithms at the same time (this is called the portfolio approach). The server contract is described by:

$$\mathsf{SATserver} = \mathtt{inst}.\textstyle\sum_i \mathtt{alg}_i.\sum_j \mathtt{par}_j.\overline{\mathtt{sol}}$$

A simple client that tries both the DPLL approach and the walksat approach can be modeled as follows:

$$\mathsf{SATclient} = \overline{\mathtt{inst}}.(\overline{\mathtt{DPLL}}.\overline{\mathtt{par}}.\mathtt{sol} + \overline{\mathtt{walksat}}.\overline{\mathtt{par}}.\mathtt{sol})$$

$$
\begin{array}{rcl}
 & \overline{\texttt{inst}}.(\overline{\texttt{DPLL}}.\texttt{sol} + \overline{\texttt{walksat}}.\texttt{sol}) & \| \quad \texttt{inst}.\sum_i \texttt{alg}_i.\overline{\texttt{sol}} \\
\longrightarrow & \overline{\texttt{inst}} \,@\, (\overline{\texttt{DPLL}}.\texttt{sol} + \overline{\texttt{walksat}}.\texttt{sol}) & \| \quad \texttt{inst} \,@\, \sum_i \texttt{alg}_i.\overline{\texttt{sol}} \\
\longrightarrow & \begin{array}{r} \overline{\texttt{inst}} \,@\, \overline{\texttt{DPLL}} \,@\, \texttt{sol} \\ \mid \overline{\texttt{inst}} \,@\, \overline{\texttt{walksat}}.\texttt{sol} \end{array} & \| \quad \begin{array}{l} \texttt{inst} \,@\, \texttt{DPLL} \,@\, \overline{\texttt{sol}} \\ \mid \texttt{inst} \,@\, \sum_{\{i \mid \texttt{A}_i \neq \texttt{DPLL}\}} \texttt{alg}_i.\overline{\texttt{sol}} \end{array} \\
\longrightarrow & \begin{array}{r} \overline{\texttt{inst}} \,@\, \overline{\texttt{DPLL}} \,@\, \texttt{sol} \\ \mid \texttt{inst} \,@\, \overline{\texttt{walksat}} \,@\, \texttt{sol} \end{array} & \| \quad \begin{array}{l} \texttt{inst} \,@\, \texttt{DPLL} \,@\, \overline{\texttt{sol}} \\ \mid \texttt{inst} \,@\, \texttt{walksat} \,@\, \overline{\texttt{sol}} \\ \mid \texttt{inst} \,@\, \sum_{\{i \mid \texttt{A}_i \neq \texttt{DPLL},\texttt{walksat}\}} \texttt{alg}_i.\overline{\texttt{sol}} \end{array} \\
\longrightarrow & \begin{array}{r} \overline{\texttt{inst}} \,@\, \overline{\texttt{DPLL}} \,@\, \texttt{sol} \\ \mid \overline{\texttt{inst}} \,@\, \overline{\texttt{walksat}} \,@\, \texttt{sol} \,@\, \mathbf{1} \end{array} & \| \quad \begin{array}{l} \texttt{inst} \,@\, \texttt{DPLL} \,@\, \overline{\texttt{sol}} \\ \mid \texttt{inst} \,@\, \texttt{walksat} \,@\, \overline{\texttt{sol}} \,@\, \mathbf{1} \\ \mid \texttt{inst} \,@\, \sum_{\{i \mid \texttt{A}_i \neq \texttt{DPLL},\texttt{walksat}\}} \texttt{alg}_i.\overline{\texttt{sol}} \end{array}
\end{array}
$$

Figure 3: An example of speculative interaction

A sample computation proceeds as described in Figure 3, assuming that the server supports both DPLL and walksat. To keep the example simple we drop the choice of parameters. Let us see in more details how the creation of threads is managed. The first reduction in Figure 3 is due to rule (*comm*), since

$$\overline{\texttt{inst}}.(\overline{\texttt{DPLL}}.\texttt{sol} + \overline{\texttt{walksat}}.\texttt{sol}) \xrightarrow{\overline{\texttt{inst}}} \overline{\texttt{inst}} \,@\, (\overline{\texttt{DPLL}}.\texttt{sol} + \overline{\texttt{walksat}}.\texttt{sol})$$

and

$$\texttt{inst}.\sum_i \texttt{alg}_i.\overline{\texttt{sol}} \xrightarrow{\texttt{inst}} \texttt{inst} \,@\, \sum_i \texttt{alg}_i.\overline{\texttt{sol}}.$$

The second reduction is also due to rule (*comm*), since, on the client side

$$\dfrac{\dfrac{}{\overline{\texttt{DPLL}}.\texttt{sol} + \overline{\texttt{walksat}}.\texttt{sol} \xrightarrow{\overline{\texttt{DPLL}},\, \overline{\texttt{walksat}}.\texttt{sol}} \overline{\texttt{DPLL}} \,@\, \texttt{sol}}\,(\textsc{Fork})}{\overline{\texttt{inst}} \,@\, (\overline{\texttt{DPLL}}.\texttt{sol} + \overline{\texttt{walksat}}.\texttt{sol}) \xrightarrow{\overline{\texttt{inst}}\,\overline{\texttt{DPLL}},\, \overline{\texttt{inst}} \,@\, \overline{\texttt{walksat}}.\texttt{sol}} \overline{\texttt{inst}} \,@\, \overline{\texttt{DPLL}} \,@\, \texttt{sol}}\,(@\text{-}\alpha\text{-}T)$$

whereas, on the server side,

$$\dfrac{\dfrac{}{\sum_i \texttt{alg}_i.\overline{\texttt{sol}} \xrightarrow{\texttt{DPLL},\, \sum_{\{i \mid \texttt{A}_i \neq \texttt{DPLL}\}} \texttt{alg}_i.\overline{\texttt{sol}}} \texttt{DPLL} \,@\, \overline{\texttt{sol}}}\,(\textsc{Fork})}{\texttt{inst} \,@\, \sum_i \texttt{alg}_i.\overline{\texttt{sol}} \xrightarrow{\texttt{inst}\,\texttt{DPLL},\, \texttt{inst} \,@\, \sum_{\{i \mid \texttt{A}_i \neq \texttt{DPLL}\}} \texttt{alg}_i.\overline{\texttt{sol}}} \texttt{inst} \,@\, \texttt{DPLL} \,@\, \overline{\texttt{sol}}}\,(@\text{-}\alpha\text{-}T)$$

3. Compliance

The compliance relation for session contracts [13, 14] consists in requiring that, whenever no reduction is possible, all client's requests and offers have been satisfied, i.e. the client is in the success state $\mathbf{1}$. For retractable contracts, thanks to the retractable operational semantics taking care of forward and backward reductions, we can adopt the same definition. We use $\xrightarrow{*}$ to denote the reflexive and transitive closure of $\longrightarrow$, and $\not\longrightarrow$ to specify that no $\longrightarrow$ reduction exists.

Definition 8 (Retractable Compliance Relation $\dashv\|^{\mathsf{R}}$).

$$\frac{}{\Gamma \ \rhd \ \mathbf{1} \backsim \sigma}\ (\text{Ax}) \qquad \frac{}{\Gamma, \rho \backsim \sigma \ \rhd \ \rho \backsim \sigma}\ (\text{Hyp}) \qquad \frac{\Gamma, \alpha.\rho + \rho' \backsim \overline{\alpha}.\sigma + \sigma' \ \rhd \ \rho \backsim \sigma}{\Gamma \ \rhd \ \alpha.\rho + \rho' \backsim \overline{\alpha}.\sigma + \sigma'}\ (+ \cdot +)$$

$$\frac{\forall h \in I.\ \Gamma, \bigoplus_{i \in I} \overline{\alpha}_i.\rho_i \backsim \sum_{j \in I \cup J} \alpha_j.\sigma_j \ \rhd \ \rho_h \backsim \sigma_h}{\Gamma \ \rhd \ \bigoplus_{i \in I} \overline{\alpha}_i.\rho_i \backsim \sum_{j \in I \cup J} \alpha_j.\sigma_j}\ (\oplus \cdot +)$$

$$\frac{\forall h \in I.\ \Gamma, \sum_{j \in I \cup J} \overline{\alpha}_j.\rho_j \backsim \bigoplus_{i \in I} \alpha_i.\sigma_i \ \rhd \ \rho_h \backsim \sigma_h}{\Gamma \ \rhd \ \sum_{j \in I \cup J} \overline{\alpha}_j.\rho_j \backsim \bigoplus_{i \in I} \alpha_i.\sigma_i}\ (+ \cdot \oplus)$$

Figure 4: System $\rhd$

i) The relation $\dashv\!\!\mid^{\mathsf{R}}$ on contracts with history is defined by:
$\mathrm{H}_1 \ltimes \rho \dashv\!\!\mid^{\mathsf{R}} \mathrm{H}_2 \ltimes \sigma$ if, for any $\mathrm{H}'_1, \mathrm{H}'_2, \rho', \sigma'$ such that
$\mathrm{H}_1 \ltimes \rho \parallel \mathrm{H}_2 \ltimes \sigma \xrightarrow{*} \mathrm{H}'_1 \ltimes \rho' \parallel \mathrm{H}'_2 \ltimes \sigma' \not\longrightarrow$, we have $\rho' = \mathbf{1}$

ii) The relation $\dashv\!\!\mid^{\mathsf{R}}$ on contracts is defined by: $\rho \dashv\!\!\mid^{\mathsf{R}} \sigma$ if $\langle\,\rangle \ltimes \rho \dashv\!\!\mid^{\mathsf{R}} \langle\,\rangle \ltimes \sigma$.

For speculative contracts we need to take into account the fact that the whole computation succeeds if at least one of its branches succeeds.

Definition 9 (Speculative Compliance Relation $\dashv\!\!\mid^{\mathsf{S}}$).
The relation $\dashv\!\!\mid^{\mathsf{S}}$ on contracts is defined by:
$\rho \dashv\!\!\mid^{\mathsf{S}} \sigma$ if for any $\mathbf{C}_\rho, \mathbf{C}_\sigma$ such that $\rho \parallel \sigma \xrightarrow{*} \mathbf{C}_\rho \parallel \mathbf{C}_\sigma \not\longrightarrow$
there exist $\mathbf{C}, n, \alpha_1, \ldots, \alpha_n$ such that $\mathbf{C}_\rho = \mathbf{C} \mid \alpha_1 @ \ldots @ \alpha_n @ \mathbf{1}$

We now provide a formal system characterizing compliance on both retractable and speculative contracts.

Definition 10 (Formal System for Compliance $\rhd$).
Judgments in the formal system $\rhd$ are expressions of the form $\Gamma \rhd \rho \backsim \sigma$, where the environment Γ is a finite set of expressions of the form $\delta \backsim \gamma$, with $\rho, \sigma, \delta, \gamma \in \mathbf{rs}\mathsf{C}$. Axioms and rules are defined in Figure 4.

Intuitively, the symbol $\backsim$ is used as syntactical counterpart for both the relations $\dashv\!\!\mid^{\mathsf{R}}$ and $\dashv\!\!\mid^{\mathsf{S}}$ that, as we prove below, do coincide.

The only non standard rule of system $\rhd$ is $(+ \cdot +)$, which ensures compliance of two external choices when they contain respectively (at least) *one* α and the corresponding $\overline{\alpha}$, followed by compliant contracts. This contrasts with the rules $(\oplus \cdot +)$ and $(+ \cdot \oplus)$, where *each* α in an internal choice must have a corresponding $\overline{\alpha}$ in the external choice, followed by compliant contracts. No rule is provided for the case $(\oplus \cdot \oplus)$ since two internal choices are compliant only if both of them are unary choices (otherwise they may always get stuck by choosing incompatible actions). Since unary internal choice coincides with unary external choice, this

Prove$(\Gamma \rhd \rho \curlywedge \sigma) =$

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

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

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.** $\mathcal{D} = \mathbf{Prove}(\Gamma, \rho \curlywedge \sigma \rhd \rho_k \curlywedge \sigma_k) \neq \mathbf{fail}$

then $\dfrac{\mathcal{D}}{\Gamma \rhd \rho \curlywedge \sigma}\,(+ \cdot +)$ **else fail**

else if $\rho = \bigoplus_{i \in I} \overline{\alpha}_i.\rho_i$ **and** $\sigma = \sum_{j \in I \cup J} \alpha_j.\sigma_j$

and for all $k \in I$ $\mathcal{D}_k = \mathbf{Prove}(\Gamma, \rho \curlywedge \sigma \rhd \rho_k \curlywedge \sigma_k) \neq \mathbf{fail}$

then $\dfrac{(\forall k \in I)\ \ \mathcal{D}_k}{\Gamma \rhd \rho \curlywedge \sigma}\,(\oplus \cdot +)$

else if $\rho = \sum_{j \in I \cup J} \overline{\alpha}_j.\rho_j$ **and** $\sigma = \bigoplus_{i \in I} \alpha_i.\sigma_i$

and for all $k \in I$ $\mathcal{D}_k = \mathbf{Prove}(\Gamma, \rho \curlywedge \sigma \rhd \rho_k \curlywedge \sigma_k) \neq \mathbf{fail}$

then $\dfrac{(\forall k \in I)\ \ \mathcal{D}_k}{\Gamma \rhd \rho \curlywedge \sigma}\,(+ \cdot \oplus)$ **else fail**

else fail

Figure 5: The procedure **Prove**.

case is taken into account by the rules we already have. Notice that rule $(+ \cdot +)$ implicitly represents the fact that, in the decision procedure for two contracts made of external choices, the possible synchronizing branches have to be tried, until either a successful one is found or all fail. Looking at a derivation bottom-up, at each application of a rule the considered pair of contracts is added to the environment Γ. In this way, if the same pair is reached again due to the equi-recursive view of contracts, the derivation can be closed using rule (Hyp). Rule (Ax) instead closes the derivation when the client reaches the success state $\mathbf{1}$. We write $\rhd \rho \curlywedge \sigma$ instead of $\Gamma \rhd \rho \curlywedge \sigma$ when Γ is empty.

Derivability in system $\rhd$ is decidable, since it is syntax-directed and proof reconstruction does terminate.
The procedure **Prove** in Figure 5 clearly implements the formal system and can be used to prove decidability of $\rhd$ (see Appendix A.1).

Theorem 1. *Derivability in the formal system $\rhd$ is decidable.*

We can now prove (see Appendix A.1) the soundness and the completeness of the formal system $\rhd$ w.r.t. both the retractable and the speculative semantics.

Theorem 2 (Retractable Soundness and Completeness)**.**

$$\rhd \rho \curlywedge \sigma \quad \textit{iff} \quad \rho \dashv\!\shortmid^{\mathsf{R}} \sigma$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\gamma_1,\gamma_2,\gamma_3 \ \rhd\ \mathbf{1} \backsim \mathbf{1}}\,(\text{Ax})}{\gamma_1,\gamma_2 \ \rhd\ \mathtt{sol} \backsim \overline{\mathtt{sol}}}\,(+\cdot\oplus)}{\gamma_1 \ \rhd\ \overline{\mathtt{DPLL}}.\mathtt{sol} + \overline{\mathtt{walksat}}.\mathtt{sol} \backsim \sum_i \mathtt{alg}_i.\overline{\mathtt{sol}}}\,(+\cdot+)}{\rhd\ \overline{\mathtt{inst}}.(\overline{\mathtt{DPLL}}.\mathtt{sol} + \overline{\mathtt{walksat}}.\mathtt{sol}) \backsim \mathtt{inst}.\sum_i \mathtt{alg}_i.\overline{\mathtt{sol}}}\,(\oplus\cdot+)$$

where $\gamma_1 = \overline{\mathtt{inst}}.(\overline{\mathtt{DPLL}}.\mathtt{sol} + \overline{\mathtt{walksat}}.\mathtt{sol}) \backsim \mathtt{inst}.\sum_i \mathtt{alg}_i.\overline{\mathtt{sol}}$
$\gamma_2 = \overline{\mathtt{DPLL}}.\mathtt{sol} + \overline{\mathtt{walksat}}.\mathtt{sol} \backsim \sum_i \mathtt{alg}_i.\overline{\mathtt{sol}}$
$\gamma_3 = \mathtt{sol} \backsim \overline{\mathtt{sol}}$
and where, for some i, $\mathtt{alg}_i = \mathtt{walksat}$.

Figure 6: A sample derivation in $\rhd$

Theorem 3 (Speculative Soundness and Completeness)**.**

$$\rhd\ \rho \backsim \sigma \quad \textit{iff} \quad \rho \dashv\!\|^{\mathsf{S}}\ \sigma$$

The client and the server of Example 3 can be now shown to be compliant, i.e. $\overline{\mathtt{inst}}.(\overline{\mathtt{DPLL}}.\mathtt{sol} + \overline{\mathtt{walksat}}.\mathtt{sol}) \dashv\!\|^{\mathsf{S}} \mathtt{inst}.\sum_i \mathtt{alg}_i.\overline{\mathtt{sol}}$, by providing a derivation for $\rhd\ \overline{\mathtt{inst}}.(\overline{\mathtt{DPLL}}.\mathtt{sol} + \overline{\mathtt{walksat}}.\mathtt{sol}) \backsim \mathtt{inst}.\sum_i \mathtt{alg}_i.\overline{\mathtt{sol}}$, as shown in Figure 6.

By the soundness and completeness of system $\rhd$ w.r.t. both the relations of retractable and speculative compliance, we immediately get that the two compliance relations do coincide.

Corollary 4 (Retractable and Speculative Compliances Coincide)**.**

$$\dashv\!\|^{\mathsf{R}} \quad = \quad \dashv\!\|^{\mathsf{S}}$$

By the above, from now on we write $\dashv\!\|$ instead of $\dashv\!\|^{\mathsf{R}}$ or $\dashv\!\|^{\mathsf{S}}$. So the following also easily follows.

Corollary 5 (Compliance Decidability)**.** *The relation* $\dashv\!\|$ *is decidable.*

Remark 3. We now discuss the impact on the compliance relation of the four mechanisms for controlling reversibility in the retractable semantics (see Remark 2). In particular, we analyze what would happen by dropping each one of them in isolation:

Dropping "Only external choices are retractable":
each reduction could be undone. From the compliance point of view, all the choices would be retractable. Hence, retractable contracts would not be a conservative extension (see Section 3.1) of session contracts any more. The case we consider is strictly more general, since we allow for both retractable and unretractable choices, being our internal choices unretractable.

Dropping the side condition $\rho \neq \mathbf{1}$ in rule (rbk) of Definition 4: any forward finite interaction would be followed by a rollback. In particular, most of the client/server pairs without recursion (except a few trivial ones, like $\langle\,\rangle \ltimes \mathbf{1} \parallel \langle\,\rangle \ltimes \sigma$) would end into $\langle\,\rangle \ltimes \circ \parallel \langle\,\rangle \ltimes \circ$. Thus all these pairs of contracts would not be compliant.

Dropping "rule (rbk) can be applied only if no other rule applies": interactions could rollback before succeeding. As in the case above, most client/server pairs (except a few trivial ones, but including recursive ones) could reduce to $\langle\,\rangle \ltimes \circ \parallel \langle\,\rangle \ltimes \circ$. Again all these pairs of contracts would not be compliant.

Dropping "in choices the selected path is not stored in the history": any client/server pair that would not normally succeed with at least one retractable choice could diverge by undoing and redoing the choice forever, thus trivially ensuring compliance.

None of the last three scenarios provides a reasonable setting. The first one would be reasonable, but too restrictive, since it would prevent to model all those irrevocable choices which are present in everyday interactions.

3.1. Conservativity

It is possible to show that all the relations on our retractable and speculative contracts (**rs**C) are conservative extensions of corresponding notions on (first-order) session contracts (SC) as defined in [13, 14], and on the retractable session contracts (**r**C) as defined in [18].

For the sake of completeness and readability, in the following subsection we briefly recall the formalism of session contracts together with some of their properties, while referring to [18] for retractable session contracts.

3.1.1. Session Contracts

The set SC of session contracts (roughly interpreting session types [4] into contracts [1, 2, 3]) can be seen as the subset of elements in **rs**C not containing external output choices and internal input choices, with the following operational semantics.

Definition 11 (Semantics of Session Contracts).

$$\overline{a}.\sigma \oplus \sigma' \xrightarrow{\tau}_{\mathsf{SC}} \overline{a}.\sigma \qquad \alpha.\sigma \xrightarrow{\alpha}_{\mathsf{SC}} \sigma \qquad a.\sigma + \sigma \xrightarrow{a}_{\mathsf{SC}} \sigma$$

As done for **rs**C (see Section 2), we write $\alpha_k.\sigma_k + \sigma'$ for $\sum_{i\in I} \alpha_i.\sigma_i$ where $k \in I$ and $\sigma' = \sum_{i\in(I\setminus\{k\})} \alpha_i.\sigma_i$ (and similarly for internal choices). Moreover, we can look at session contracts up-to unfolding of recursion.

The next definitions introduce the LTS for client/server pairs of session contracts, and the corresponding compliance relation.

Definition 12 (Semantics of Client/Server Pairs of Session Contracts).

$$\frac{\rho \xrightarrow{\tau}_{\mathsf{SC}} \rho'}{\rho\|\sigma \longrightarrow_{\mathsf{SC}} \rho'\|\sigma} \qquad \frac{\sigma \xrightarrow{\tau}_{\mathsf{SC}} \sigma'}{\rho\|\sigma \longrightarrow_{\mathsf{SC}} \rho\|\sigma'} \qquad \frac{\rho \xrightarrow{\alpha}_{\mathsf{SC}} \rho' \qquad \sigma \xrightarrow{\overline{\alpha}}_{\mathsf{SC}} \sigma'}{\rho\|\sigma \longrightarrow_{\mathsf{SC}} \rho'\|\sigma'}$$

$$\frac{}{\Gamma \ \rhd_{\mathsf{SC}} \mathbf{1} \backsim \sigma}\ (\textsc{Ax}) \qquad \frac{}{\Gamma, \rho \backsim \sigma \ \rhd_{\mathsf{SC}} \rho \backsim \sigma}\ (\textsc{Hyp})$$

$$\frac{\forall h \in I.\ \Gamma, \bigoplus_{i\in I} \overline{a}_i.\rho_i \backsim \sum_{j\in I\cup J} a_j.\sigma_j \ \rhd_{\mathsf{SC}} \rho_h \backsim \sigma_h}{\Gamma \ \rhd_{\mathsf{SC}} \bigoplus_{i\in I} \overline{a}_i.\rho_i \backsim \sum_{j\in I\cup J} a_j.\sigma_j}\ (\oplus \cdot +)$$

$$\frac{\forall h \in I.\ \Gamma, \sum_{j\in I\cup J} a_j.\rho_j \backsim \bigoplus_{i\in I} \overline{a}_i.\sigma_i \ \rhd_{\mathsf{SC}} \rho_h \backsim \sigma_h}{\Gamma \ \rhd_{\mathsf{SC}} \sum_{j\in I\cup J} a_j.\rho_j \backsim \bigoplus_{i\in I} \overline{a}_i.\sigma_i}\ (+ \cdot \oplus)$$

Figure 7: System $\rhd_{\mathsf{SC}}$

Definition 13 (Compliance Relation $\dashv\!\shortparallel_{\mathsf{SC}}$ for Session Contracts).
The relation $\dashv\!\shortparallel_{\mathsf{SC}} \subset \mathsf{SC} \times \mathsf{SC}$ is defined by:
$\rho \dashv\!\shortparallel_{\mathsf{SC}} \sigma$ if, for each ρ', σ' such that
$\rho \parallel \sigma \xrightarrow{*}_{\mathsf{SC}} \rho' \parallel \sigma' \not\rightarrow_{\mathsf{SC}}$ we have $\rho' = \mathbf{1}$

The sound and complete formal system $\rhd_{\mathsf{SC}}$ for $\dashv\!\shortparallel_{\mathsf{SC}}$ is recalled in Figure 7.

Theorem 6. *Let $\rho, \sigma \in \mathsf{SC}$. $\rho \dashv\!\shortparallel_{\mathsf{SC}} \sigma$ iff $\rhd_{\mathsf{SC}} \rho \backsim \sigma$*

Proof sketch. In [13] a formal system $\rhd_{\mathsf{H}}$ is devised which is sound and complete for compliance of higher-order session contracts (HSC), that is $\rhd_{\mathsf{H}} \rho \backsim \sigma$ iff $\rho \dashv\!\shortparallel_{\mathsf{H}} \sigma$. The set SC can be looked at as the first-order restriction of HSC. It is easy to show that, for $\rho, \sigma \in \mathsf{SC}$, $\rhd_{\mathsf{H}} \rho \backsim \sigma$ iff $\rhd_{\mathsf{SC}} \rho \backsim \sigma$.
It is also not difficult to show that, for $\rho, \sigma \in \mathsf{SC}$, $\rho \dashv\!\shortparallel_{\mathsf{SC}} \sigma$ iff $\rho \dashv\!\shortparallel_{\mathsf{H}} \sigma$.
From the above statement, the thesis descends immediately. □

3.1.2. Conservativity Results

As previously said, it is not difficult to check that session contracts SC are a strict subset of retractable session contracts $\mathbf{rC}$, which, in turn, are a strict subset of the contracts $\mathbf{rsC}$ we are presently investigating, namely: $\mathsf{SC} \subsetneq \mathbf{rC} \subsetneq \mathbf{rsC}$. Obviously the strict inclusion $\mathsf{SC} \subsetneq \mathbf{rsC}$ is not enough, by itself, to guarantee the retractable and speculative operational semantics for $\mathbf{rsC}$ to be conservative extensions of the operational semantics of SC. We show that it is so in the following Proposition 7 (see Appendix A.1.1 for the proof). Informally, it states that both the forward retractable semantics $\longrightarrow_f$ and the speculative semantics $\longrightarrow$ of pairs of contracts in SC are annotated versions of their semantics as session contracts.

Proposition 7 (Operational Semantics Conservativity)**.** *Let $\rho, \sigma \in \mathsf{SC}$.*

i) $\rho \parallel \sigma \xrightarrow{*}_{\mathsf{SC}} \rho' \parallel \sigma'$ *iff for all* $\mathrm{H}_1, \mathrm{H}_2$ *there exist* $\mathrm{H}_1', \mathrm{H}_2'$ *such that* $\mathrm{H}_1 \ltimes \rho \parallel \mathrm{H}_2 \ltimes \sigma \xrightarrow{*}_f \mathrm{H}_1' \ltimes \rho' \parallel \mathrm{H}_2' \ltimes \sigma'$

ii) $\rho \parallel \sigma \xrightarrow{*}_{\mathsf{SC}} \rho' \parallel \sigma'$ *iff there exist* $n, \alpha_1, \ldots, \alpha_n, \mathbf{C}_\rho$ *and* $\mathbf{C}_\sigma$ *such that* $\rho \parallel \sigma \xrightarrow{*} \alpha_1 @ \ldots \alpha_n @ \rho' \mid \mathbf{C}_\rho \parallel \overline{\alpha_1} @ \ldots \overline{\alpha_n} @ \sigma' \mid \mathbf{C}_\sigma$

where $\longrightarrow_{\mathsf{SC}}$ *denotes the reduction relation on* SC *client/server pairs in the theory of session contracts.*

We do not take into account conservativity of the retractable operational semantics for **rs**C over the one for **r**C because it is quite trivial, since the rules in the two semantics are essentially the same. A conservativity result of the speculative operational semantics for **rs**C over the one for **r**C would instead consist in a rather cumbersome and uninteresting statement.

The conservativity result for the operational semantics is not enough, in itself, to guarantee the theory of retractable compliance for **rs**C to be a conservative extension of both the theory of compliance for **r**C and for SC. Also in this case, however, we can prove it to be so, that is compliance for session contracts is the restriction of our relation $\dashv\!\!\|$ to elements in SC. Similarly, compliance for retractable session contracts is the restriction of $\dashv\!\!\|$ to elements in **r**C.
To prove such results, let $\dashv\!\!\|_{\mathbf{rC}}$ and $\rhd_{\mathbf{rC}}$ be, respectively, the compliance relations on retractable session contracts and the formal system axiomatizing it (see [18]).

We first show that the logical theories of $\rhd_{\mathsf{SC}}$ and $\rhd_{\mathbf{rC}}$ are conservative extensions of the logical theory $\rhd$.

Proposition 8 (Formal Systems Conservativity)**.**

i) Let $\rho, \sigma \in \mathsf{SC}$*:* $\quad \rhd_{\mathsf{SC}} \rho \backsim\!\!\dashv \sigma \quad$ *iff* $\quad \rhd\ \rho \backsim\!\!\dashv \sigma$

ii) Let $\rho, \sigma \in \mathbf{rC}$*:* $\quad \rhd_{\mathbf{rC}} \rho \backsim\!\!\dashv \sigma \quad$ *iff* $\quad \rhd\ \rho \backsim\!\!\dashv \sigma$

Proof sketch. By inspection of the rules of the formal systems, and by $\mathsf{SC} \subsetneq \mathbf{rC} \subsetneq \mathbf{rsC}$. □

From Proposition 8 and the soundness and completeness property of $\rhd_{\mathsf{SC}}$ (Theorem 6) and $\rhd_{\mathbf{rC}}$ [18, Theorems 3.9 and 3.12] we immediately get what follows.

Corollary 9 (Compliances Conservativity)**.**

i) Let $\rho, \sigma \in \mathsf{SC}$*:* $\quad \rho \dashv\!\!\|_{\mathsf{SC}} \sigma \quad$ *iff* $\quad \rho \dashv\!\!\| \sigma$

ii) Let $\rho, \sigma \in \mathbf{rC}$*:* $\quad \rho \dashv\!\!\|_{\mathbf{rC}} \sigma \quad$ *iff* $\quad \rho \dashv\!\!\| \sigma$

4. Duality and the Subcontract Relation

Unlike the retractable session contracts of [18], in the present setting a natural notion of *duality* exists. The dual $\overline{\sigma}$ of an element σ of **rs**C is obtained, as for session contracts, by interchanging any name a with $\overline{a}$ and $+$ with $\oplus$.

Formally, we first define duality for (possibly open) contracts, that we dub **rs**Co, and then we restrict such a definition to **rs**C (i.e., to closed expressions).

Definition 14 (Syntactic duality).

i) Let $\sigma \in \mathbf{rsCo}$. The syntactic dual $\overline{\sigma}$ of σ is defined by the following clauses:
$\overline{\mathbf{1}} = \mathbf{1} \qquad \overline{x} = x \qquad \overline{\mathsf{rec}\, x.\sigma} = \mathsf{rec}\, x.\overline{\sigma}$
$\overline{\sum_{i\in I} \alpha_i.\sigma_i} = \bigoplus_{i\in I} \overline{\alpha}_i.\overline{\sigma}_i \qquad \overline{\bigoplus_{i\in I} \alpha_i.\sigma_i} = \sum_{i\in I} \overline{\alpha}_i.\overline{\sigma}_i$

ii) We define $\overline{(\cdot)} : \mathbf{rsC} \to \mathbf{rsC}$ as the restriction to $\mathbf{rsC}$ of the duality function on $\mathbf{rsCo}$, observing that $\overline{\sigma} \in \mathbf{rsC}$ iff $\sigma \in \mathbf{rsC}$.

From now on, in order to avoid too cumbersome definitions, any time an inductive definition on elements of $\mathbf{rsC}$ is provided, it will be tacitly assumed to be the restriction to $\mathbf{rsC}$ of the corresponding inductive definition on $\mathbf{rsCo}$.
A first relevant property of duality is that any contract is compliant with its syntactic dual.

Proposition 10. *For any $\sigma \in \mathbf{rsC}$, $\sigma \dashv\!\shortparallel \overline{\sigma}$.*

Proof. Since $\overline{\sigma}$ is obtained from σ by exchanging each α with $\overline{\alpha}$ and $+$ with $\oplus$, it is easy to get a derivation of $\rhd\ \overline{\sigma} \curlywedge \sigma$. The thesis is then an immediate consequence of soundness and completeness of $\rhd$. □

The notion of dual contract allows one to combine pairs of contracts in the compliance relation using a sort of "transitive" property, provided that the two contracts "in the middle" are dual of each other (see Appendix A.2 for the proof). Operationally, one can think at the two contracts in the middle as the specification of a bidirectional forwarder, and the proposition means that if a client and a server can correctly interact via a forwarder then they can also correctly interact directly.

Proposition 11. *For any $\rho, \sigma, \sigma' \in \mathbf{rsC}$, $\rho \dashv\!\shortparallel \sigma$ and $\overline{\sigma} \dashv\!\shortparallel \sigma'$ imply $\rho \dashv\!\shortparallel \sigma'$*

We will provide further properties of duality using the notion of subcontract relation. Indeed, the notion of compliance naturally induces a substitutability relation on servers, denoted $\preccurlyeq_s$, that we call *subcontract relation for servers*. Intuitively, larger elements in the relation are compliant with a superset of the clients of smaller elements. Such a relation may be used for implementing contract-based query engines (see [30] for a detailed discussion). An analogous subcontract relation, denoted $\preccurlyeq_c$, can be defined for clients.

Definition 15 (Subcontract Relations for Servers and for Clients).
Let $\sigma, \sigma' \in \mathbf{rsC}$. We define

i) $\sigma \preccurlyeq_s \sigma' \triangleq \forall \rho \in \mathbf{rsC}\ [\, \rho \dashv\!\shortparallel \sigma \text{ implies } \rho \dashv\!\shortparallel \sigma' \,]$

ii) $\sigma \preccurlyeq_c \sigma' \triangleq \forall \rho \in \mathbf{rsC}\ [\, \sigma \dashv\!\shortparallel \rho \text{ implies } \sigma' \dashv\!\shortparallel \rho \,]$

Using Proposition 11 we can characterize both $\preccurlyeq_s$ and $\preccurlyeq_c$ in terms of duality and compliance, relate them and get their decidability (see Appendix A.2 for the proof).

Theorem 12. *For any $\sigma, \sigma' \in \mathbf{rsC}$:*

i) $\sigma \preccurlyeq_s \sigma'$ *iff* $\overline{\sigma} \dashv\!\shortparallel \sigma'$

ii) $\sigma \preccurlyeq_c \sigma'$ *iff* $\sigma' \dashv\!\shortparallel \overline{\sigma}$

iii) $\sigma \preccurlyeq_s \sigma'$ *iff* $\overline{\sigma'} \preccurlyeq_c \overline{\sigma}$

iv) $\sigma \preccurlyeq_s \sigma'$ *and* $\sigma \preccurlyeq_c \sigma'$ *are decidable.*

By item iii) above, from now on we can concentrate on the relation $\preccurlyeq_s$.

We can now characterize duality in terms of the subcontract relation for servers: given a client ρ, its dual $\overline{\rho}$ is a least element among all its possible servers, namely it is a possible server, and it is smaller than all the other possible servers.

Proposition 13 (Dual as a Least Element w.r.t. $\preccurlyeq_s$)**.**
Let $\rho \in \mathbf{rsC}$*. Then* $\overline{\rho}$ *is a server for* ρ*, namely* $\rho \dashv\!\shortparallel \overline{\rho}$*, and more precisely it is a least element in the set of the servers of* ρ*, that is,*

$$\forall \sigma \in \mathbf{rsC}\text{:}\quad \rho \dashv\!\shortparallel \sigma \quad \textit{implies} \quad \overline{\rho} \preccurlyeq_s \sigma$$

Proof. Suppose that $\rho \dashv\!\shortparallel \sigma$ and take any contract δ such that $\delta \dashv\!\shortparallel \overline{\rho}$. Since $\overline{\overline{\rho}} = \rho$, by Proposition 11 we know that $\delta \dashv\!\shortparallel \sigma$; hence $\overline{\rho} \preccurlyeq_s \sigma$ by definition. □

Since we have not yet proved that the subcontract relation is a partial order, we do not know yet whether $\overline{\rho}$ is also a minimal, i.e. there is no smaller element, neither whether other least elements or minimal elements exist. These questions will be answered by Proposition 17.

As done for the compliance relation, we characterize now the subcontract relation for servers in terms of derivability in the following formal system, where the symbol $\ll$ is used as syntactical counterpart of the relation $\preccurlyeq_s$.

Definition 16 (Formal System for Subcontract ▶).
Judgments in the formal system ▶ are expressions of the form $\Gamma \blacktriangleright \rho \ll \sigma$, where the environment Γ is a finite set of expressions of the form $\delta \ll \gamma$, with $\rho, \sigma, \delta, \gamma \in \mathbf{rsC}$. Axioms and rules are defined in Figure 8.

The rules in system ▶ can be read as a translation of the rules in system $\rhd$ (see Figure 4) via Theorem 12(i). As for $\rhd$, in $\Gamma \blacktriangleright \rho \ll \sigma$ we may drop Γ if empty.

Lemma 14. $\Gamma \blacktriangleright \sigma \ll \sigma'$ *iff* $\widetilde{\Gamma} \rhd \overline{\sigma} \backsim\!\!\mid \sigma'$
where $\Gamma = \{\sigma_i \ll \sigma'_i\}_{i \in I}$ *and* $\widetilde{\Gamma} = \{\overline{\sigma_i} \backsim\!\!\mid \sigma'_i\}_{i \in I}$.

Proof. ($\Rightarrow$) By induction over the derivation of $\Gamma \blacktriangleright \sigma \ll \sigma'$.
($\Leftarrow$) By induction over the derivation of $\widetilde{\Gamma} \rhd \sigma \backsim\!\!\mid \sigma'$. □

System ▶ is sound and complete for the subcontract relation $\preccurlyeq_s$.

Theorem 15 (Soundness and Completeness of ▶)**.** $\blacktriangleright \sigma \ll \sigma'$ *iff* $\sigma \preccurlyeq_s \sigma'$

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

$$\frac{\Gamma, \alpha.\sigma_1 \oplus \sigma_2 \ll \alpha.\sigma'_1 + \sigma'_2 \ \blacktriangleright \ \sigma_1 \ll \sigma'_1}{\Gamma \ \blacktriangleright \ \alpha.\sigma_1 \oplus \sigma_2 \ll \alpha.\sigma'_1 + \sigma'_2}\ (\oplus \cdot + \text{-}\preccurlyeq_s)$$

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

$$\frac{\forall h \in I.\ \ \Gamma, \bigoplus_{j\in I\cup J}\alpha_j.\sigma_j \ll \bigoplus_{i\in I}\alpha_i.\sigma'_i \ \blacktriangleright \ \sigma_h \ll \sigma'_h}{\Gamma \ \blacktriangleright \ \bigoplus_{j\in I\cup J}\alpha_j.\sigma'_j \ll \bigoplus_{i\in I}\alpha_i.\sigma'_i}\ (\oplus \cdot \oplus \text{-}\preccurlyeq_s)$$

Figure 8: The formal system $\blacktriangleright$

Proof. ($\Rightarrow$) Let $\blacktriangleright \ \sigma \ll \sigma'$. By Lemma 14 we get $\rhd \ \overline{\sigma} \backsim \sigma'$ and hence $\overline{\sigma} \dashv\!\shortparallel \sigma'$ by soundness of system $\rhd$. The thesis now descends from Theorem 12.
($\Leftarrow$) Let $\sigma \preccurlyeq_s \sigma'$. By Theorem 12 we have that $\overline{\sigma} \dashv\!\shortparallel \sigma'$. By completeness of system $\rhd$ we get $\rhd \ \overline{\sigma} \backsim \sigma'$. Now, by Lemma 14, we can obtain $\blacktriangleright \ \sigma \ll \sigma'$. □

System $\blacktriangleright$ can be used to show that $\preccurlyeq_s$ is antisymmetric (see Appendix A.2) and hence that it is a partial order. Moreover $\overline{\rho}$ is also the minimum server of ρ: it is minimal, hence there is no smaller server, and there is a unique minimal.

Proposition 16. *The relation $\preccurlyeq_s$ is antisymmetric. That is, for any $\sigma, \sigma' \in \mathbf{rsC}$,*

$$\sigma \preccurlyeq_s \sigma' \text{ and } \sigma' \preccurlyeq_s \sigma \text{ implies } \sigma = \sigma'$$

Proposition 17. *$\preccurlyeq_s$ is a partial order $\wedge$ $\forall \rho \in \mathbf{rsC}$, $\overline{\rho}$ is the minimum server of ρ.*

Proof sketch. We need to show $\preccurlyeq_s$ to be reflexive, transitive and antisymmetric. Reflexivity and transitivity immediately descend from the definition of $\preccurlyeq_s$ (Definition 15). Antisymmetry follows from Proposition 16.

For the second conjunct of the statement, suppose, towards a contradiction, that there exists $\sigma \neq \overline{\rho}$ such that $\rho \dashv\!\shortparallel \sigma$ and $\sigma \preccurlyeq_s \overline{\rho}$. By Proposition 13 we have $\overline{\rho} \preccurlyeq_s \sigma$. By antisymmetry we have $\sigma = \overline{\rho}$, against the hypothesis. □

The structure of the partial order is shown in Figure 9, where the relations between terms with a unique choice among actions a, b, c, $\overline{a}$, $\overline{b}$ and $\overline{c}$ are pictured. Notably, duality can be seen graphically as central inversion.

Remark 4. Analogously to what done in Section 3.1, one can show the subcontract relation $\preccurlyeq_s$ to be a conservative extension of the corresponding notion in SC. Moreover, the restriction of $\preccurlyeq_s$ to $\mathbf{rC}$ provides a suitable notion of subcontract for $\mathbf{rC}$ (which has never been studied before).

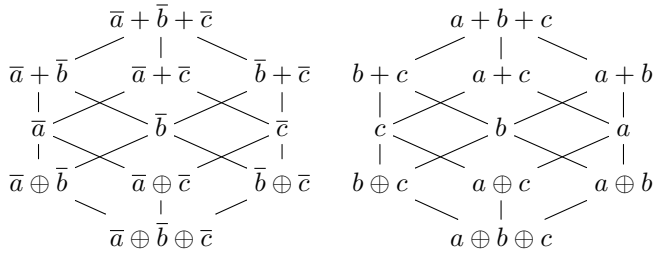

Figure 9: Subcontract preorder: a sample

5. Complexity Issues

The decision algorithm **Prove** for compliance of Figure 5 is simple, but, as it is, its complexity is exponential. The example below shows how an exponential number of recursive calls can be actually reached. It is an adaptation of the example presented in [31](§11) for the subtyping relation for recursive arrow and product types.

For each $n \in \mathbb{N}$ we define two contracts ρ_n and σ_n by induction, as follows.

$$\begin{array}{rclcrcl} \rho_0 & = & \mathsf{rec}\, x.(a.x + b.x) & \qquad & \rho_{n+1} & = & \mathsf{rec}\, x.(a.x + b.\rho_n) \\ \sigma_0 & = & \mathsf{rec}\, x.(\overline{a}.x \oplus \overline{b}.x) & & \sigma_{n+1} & = & \mathsf{rec}\, x.(\overline{a}.\sigma_n \oplus \overline{b}.x) \end{array}$$

As for the example in [31], the size of ρ_n and σ_n is linear in n, since ρ_n and σ_n appear just once in the definitions of ρ_{n+1} and σ_{n+1}, respectively.

By complete induction over n it is possible to prove that

Fact 18. *For any $n \in \mathbb{N}$, $\rho_n \dashv\!\shortmid \sigma_n$.*

In particular, the computation of $\mathbf{Prove}(\emptyset \rhd \rho_n \prec\!\shortmid \sigma_n)$ builds a derivation for $\rhd \rho_n \prec\!\shortmid \sigma_n$ in an actual exponential number of calls. In order to show that, let us first notice that any call of the shape $\mathbf{Prove}(\Gamma \rhd \rho_k \prec\!\shortmid \sigma_k)$, with $k \neq 0$ and $\rho_k \prec\!\shortmid \sigma_k \notin \Gamma$, produces two immediate calls: $\mathbf{Prove}(\Gamma_1 \rhd \rho_k \prec\!\shortmid \sigma_{k-1})$ and $\mathbf{Prove}(\Gamma_1 \rhd \rho_{k-1} \prec\!\shortmid \sigma_k)$. These latter two calls, produce, besides others, a call of $\mathbf{Prove}(\Gamma_2 \rhd \rho_{k-1} \prec\!\shortmid \sigma_{k-1})$ and one of $\mathbf{Prove}(\Gamma_3 \rhd \rho_{k-1} \prec\!\shortmid \sigma_{k-1})$, with $\Gamma_2 \neq \Gamma_3$, as it is shown in the following recursive-call tree (where we abbreviate "**Prove**" by "**Pr**").

$$\begin{array}{c} \mathbf{Pr}(\Gamma \rhd \rho_k \prec\!\shortmid \sigma_k) \\ \mathbf{Pr}(\Gamma_1 \rhd \rho_k \prec\!\shortmid \sigma_{k-1}) \qquad \mathbf{Pr}(\Gamma_1 \rhd \rho_{k-1} \prec\!\shortmid \sigma_k) \\ \mathbf{Pr}(\Gamma_2 \rhd \rho_k \prec\!\shortmid \sigma_{k-2}) \quad \underline{\mathbf{Pr}(\Gamma_2 \rhd \rho_{k-1} \prec\!\shortmid \sigma_{k-1})} \quad \underline{\mathbf{Pr}(\Gamma_3 \rhd \rho_{k-1} \prec\!\shortmid \sigma_{k-1})} \quad \mathbf{Pr}(\Gamma_3 \rhd \rho_{k-2} \prec\!\shortmid \sigma_k) \\ \ldots\ldots \quad \text{etc.} \end{array}$$

where $\Gamma_1 = \{\Gamma, \rho_k \prec\!\shortmid \sigma_k\}$
and $\quad \Gamma_2 = \{\Gamma, \rho_k \prec\!\shortmid \sigma_k, \rho_k \prec\!\shortmid \sigma_{k-1}\} \;\neq\; \{\Gamma, \rho_k \prec\!\shortmid \sigma_k, \rho_{k-1} \prec\!\shortmid \sigma_k\} = \Gamma_3$.

So, given n, the computation of $\mathbf{Pr}(\emptyset \rhd \rho_n \prec\!\shortmid \sigma_n)$ results in an overall number of *at least* 2^n recursive calls.

Nonetheless, the complexity of the compliance decision procedure can be drastically reduced down to a polynomial — actually quadratic — complexity, as shown below.

$$\frac{}{\rhd\ \mathbf{1} \rightharpoonup \sigma}\ (\mathrm{Ax}_\infty) \qquad \frac{\rho \rightharpoonup \sigma}{\alpha.\rho + \rho' \rightharpoonup \overline{\alpha}.\sigma + \sigma'}\ (+ \cdot +_\infty)$$

$$\frac{(\forall h \in I) \quad \rho_h \rightharpoonup \sigma_h}{\bigoplus_{i\in I}\overline{\alpha}_i.\rho_i \rightharpoonup \sum_{j\in I\cup J}\alpha_j.\sigma_j}\ (\oplus \cdot +_\infty) \qquad \frac{(\forall h \in I) \quad \rho_h \rightharpoonup \sigma_h}{\sum_{j\in I\cup J}\overline{\alpha}_j.\rho_j \rightharpoonup \bigoplus_{i\in I}\alpha_i.\sigma_i}\ (+ \cdot \oplus_\infty)$$

Figure 10: The non-well founded system $\rhd_\infty$

A quadratic decision algorithm.
We first define a non-well founded version of system $\rhd$, that we dub $\rhd_\infty$.

Definition 17 (The non-well founded system $\rhd_\infty$). We write $\rhd_\infty\ \rho \rightharpoonup \sigma$ whenever there exists a finite or infinite derivation tree formed by the rules in Figure 10 having $\rho \rightharpoonup \sigma$ as conclusion, and such that each finite branch ends with an instance of axiom (Ax_∞).

Systems $\rhd$ and $\rhd_\infty$ are equivalent (see Appendix A.3 for the proof).

Lemma 19. $\rhd\ \rho \rightharpoonup \sigma$ *iff* $\rhd_\infty\ \rho \rightharpoonup \sigma$

In Figure 11 we present a decision algorithm $\mathbf{Decide}_{\dashv\|}$, based on the procedures $\mathbf{P}$ and $\mathbf{P}^+$. Whereas the procedure **Prove** returns, if any, a derivation in $\rhd$, $\mathbf{Decide}_{\dashv\|}$ just checks for the existence of a derivation in $\rhd_\infty$ (and hence in $\rhd$ by Lemma 19). Its execution resembles that of an alternating Turing machine, where nodes corresponding to rules $(\oplus \cdot +_\infty)$ and $(+ \cdot \oplus_\infty)$ are universal and nodes corresponding to $(+ \cdot +_\infty)$ are existential; $\mathbf{P}(\mathsf{A}, \mathsf{F}, \mathsf{L}, \mathsf{b})$ attempts to prove all statements in its goal list L, while $\mathbf{P}^+(\mathsf{A}, \mathsf{F}, \mathsf{L}, \mathsf{b})$ succeeds if at least one goal in L is satisfiable.

The procedure $\mathbf{P}$ is an adaptation of the concrete subtyping algorithm for recursive arrow and product types of [31](§10) to the present, more complex context, where $\sum$ and $\bigoplus$ can be looked at as constructors of arbitrary arity. It consists of a proof reconstruction procedure for $\rhd_\infty$ using a *depth-first* technique. $\mathbf{P}$ accumulates in its first argument A all the judgments it encounters during the search, in order to avoid looping over the same judgments (a role similar to Γ in system $\rhd$). With respect to the algorithm in [31](§10) we have two further parameters, F and b. The argument F accumulates the judgments for which it has been found that no derivation exists. When a rule $(+ \cdot +_\infty)$ is encountered, the algorithm proceeds by calling the procedure $\mathbf{P}^+$ which, in case a premise is unprovable, goes on checking the other premises. The negative information inferred about unprovable judgments is stored in F and it is carried along by the procedure $\mathbf{P}^+$ (as well as the positive information stored in A) in order not to duplicate work. The argument b, that can be either **ok** or **fail**, is used to record whether the last call was successful or not, and it is used by $\mathbf{P}^+$ to know whether it has to stop with success, or to check a new premise.

$\mathbf{Decide}_{\dashv\!|\!|}\ (\rho \backsim \sigma) = \mathbf{let}\ \ (\mathsf{A},\mathsf{F},\mathsf{b}) = \mathbf{P}(\emptyset,\emptyset,[\rho \backsim \sigma],\mathbf{ok})$
$\mathbf{in}\ \ \mathsf{b} = \mathbf{ok}$

where

$\mathbf{P}(\mathsf{A},\mathsf{F},[\,],\mathsf{b}) = (\mathsf{A},\mathsf{F},\mathsf{b})$

$\mathbf{P}(\mathsf{A},\mathsf{F},(\rho \backsim \sigma):\mathsf{xs},\mathsf{b}) =$

-1- $\mathbf{if}\ \ \rho = \mathbf{1}\ \ \mathbf{then}\ \ \mathbf{P}(\mathsf{A},\mathsf{F},\mathsf{xs},\mathsf{b})$

-2- $\mathbf{else\ if}\ \ \rho \backsim \sigma \in \mathsf{A}\ \ \mathbf{then}\ \ \mathbf{P}(\mathsf{A},\mathsf{F},\mathsf{xs},\mathsf{b})$

-3- $\mathbf{else\ if}\ \ \rho \backsim \sigma \in \mathsf{F}\ \ \mathbf{then}\ \ (\mathsf{A},\mathsf{F},\mathbf{fail})$

-4- $\mathbf{else\ if}\ \ \rho = \sum_{i\in I}\alpha_i.\rho_i\ \ \mathbf{and}\ \sigma = \sum_{j\in J}\overline{\alpha}_j.\sigma_j\ \mathbf{and}\ I \cap J = \{i_1,\ldots,i_n\}$

-5- $\mathbf{then}\ \ \mathbf{let}\ \ (\mathsf{A}_0,\mathsf{F}_0,\mathsf{b}_0) = \mathbf{P}^{+}(\mathsf{A}\cup\{\rho \backsim \sigma\},\mathsf{F},[\rho_{i_1} \backsim \sigma_{i_1} \ldots \rho_{i_n} \backsim \sigma_{i_n}],\mathsf{b})$

-6- $\mathbf{in}\ \ \mathbf{if}\ \ \mathsf{b}_0{=}\mathbf{fail}\ \ \mathbf{then}\ \ (\mathsf{A}_0,\mathsf{F}_0,\mathbf{fail})$

-7- $\mathbf{else}\ \ \mathbf{P}(\mathsf{A}_0,\mathsf{F}_0,\mathsf{xs},\mathsf{b}_0)$

-8- $\mathbf{else\ if}\ \ \rho = \bigoplus_{i\in I}\overline{\alpha}_i.\rho_i\ \ \mathbf{and}\ \ \sigma = \sum_{j\in I\cup J}\alpha_j.\sigma_j\ \mathbf{and}\ \ I = \{i_1,\ldots,i_n\}$

-9- $\mathbf{then}\ \ \mathbf{let}\ \ (\mathsf{A}_0,\mathsf{F}_0,\mathsf{b}_0) = \mathbf{P}(\mathsf{A}\cup\{\rho \backsim \sigma\},\mathsf{F},[\rho_{i_1} \backsim \sigma_{i_1} \ldots \rho_{i_n} \backsim \sigma_{i_n}],\mathsf{b})$

-10- $\mathbf{in}\ \ \mathbf{if}\ \ \mathsf{b}_0{=}\mathbf{fail}\ \ \mathbf{then}\ \ (\mathsf{A}_0,\mathsf{F}_0,\mathbf{fail})$

-11- $\mathbf{else}\ \ \mathbf{P}(\mathsf{A}_0,\mathsf{F}_0,\mathsf{xs},\mathsf{b}_0)$

-12- $\mathbf{else\ if}\ \ \rho = \sum_{j\in I\cup J}\overline{\alpha}_j.\rho_j\ \ \mathbf{and}\ \ \sigma = \bigoplus_{i\in I}\alpha_i.\sigma_i\ \mathbf{and}\ \ I = \{i_1,\ldots,i_n\}$

-13- $\mathbf{then}\ \ \mathbf{let}\ \ (\mathsf{A}_0,\mathsf{F}_0,\mathsf{b}_0) = \mathbf{P}(\mathsf{A}\cup\{\rho \backsim \sigma\},\mathsf{F},[\rho_{i_1} \backsim \sigma_{i_1} \ldots \rho_{i_n} \backsim \sigma_{i_n}],\mathsf{b})$

-14- $\mathbf{in}\ \ \mathbf{if}\ \ \mathsf{b}_0{=}\mathbf{fail}\ \ \mathbf{then}\ \ (\mathsf{A}_0,\mathsf{F}_0,\mathbf{fail})$

-15- $\mathbf{else}\ \ \mathbf{P}(\mathsf{A}_0,\mathsf{F}_0,\mathsf{xs},\mathsf{b}_0)$

-16- $\mathbf{else\ if}\ \ \rho = \mathsf{rec}\,x.\rho'\ \ \mathbf{then}\ \ \mathbf{P}(\mathsf{A},\mathsf{F},(\{\mathsf{rec}\,x.\rho'/x\}\rho' \backsim \sigma):\mathsf{xs},\mathsf{b})$

-17- $\mathbf{else\ if}\ \ \sigma = \mathsf{rec}\,x.\sigma'\ \ \mathbf{then}\ \ \mathbf{P}(\mathsf{A},\mathsf{F},(\rho \backsim \{\mathsf{rec}\,x.\sigma'/x\}\sigma'):\mathsf{xs},\mathsf{b})$

-18- $\mathbf{else}\ \ (\mathsf{A},\mathsf{F}\cup\{\rho \backsim \sigma\},\mathbf{fail})$

and where

$\mathbf{P}^{+}(\mathsf{A},\mathsf{F},[\rho \backsim \sigma],\mathsf{b}) = \mathbf{P}(\mathsf{A},\mathsf{F},[\rho \backsim \sigma],\mathsf{b})$

$\mathbf{P}^{+}(\mathsf{A},\mathsf{F},(\rho \backsim \sigma):\mathsf{xs},\mathsf{b}) =$

-19- $\mathbf{let}\ \ (\mathsf{A}_0,\mathsf{F}_0,\mathsf{b}_0) = \mathbf{P}(\mathsf{A},\mathsf{F},[\rho \backsim \sigma],\mathsf{b})\ \ \mathbf{in}$

-20- $\mathbf{if}\ \ \mathsf{b}_0 = \mathbf{fail}\ \ \mathbf{then}\ \ \mathbf{P}^{+}(\mathsf{A}\cup\mathsf{A}_0,\mathsf{F}\cup\mathsf{F}_0,\mathsf{xs},\mathbf{ok})$

-21- $\mathbf{else}\ \ (\mathsf{A}_0,\mathsf{F}_0,\mathsf{b}_0)$

Figure 11: The quadratic decision procedure for compliance.

Let us note that, contrary to the previous treatment, while studying the algorithm $\mathbf{Decide}_{\dashv\|}$, we abandon the equi-recursive view of recursion, and we represent a contract by a particular explicit and (possibly) recursive expression.

We now give an upper bound on the complexity of the algorithm $\mathbf{Decide}_{\dashv\|}$ (see Appendix A.3 for the proof), based on the notion of size of a contract below.

Definition 18. The size $\mathsf{size}(\sigma)$ of a contract σ is inductively defined as follows:

$$\begin{array}{rcl@{\qquad}rcl} \mathsf{size}(\mathbf{1}) & = & 1 & & & \\ \mathsf{size}(\sum_{\mathsf{i}\in\mathsf{I}} \alpha_\mathsf{i}.\sigma_\mathsf{i}) & = & \sum_{i\in I}(1+\mathsf{size}(\sigma_\mathsf{i})) & \mathsf{size}(\mathsf{x}) & = & 1 \\ \mathsf{size}(\bigoplus_{\mathsf{i}\in\mathsf{I}} \alpha_\mathsf{i}.\sigma_\mathsf{i}) & = & \sum_{i\in I}(1+\mathsf{size}(\sigma_\mathsf{i})) & \mathsf{size}(\mathsf{rec}\,\mathsf{x}.\sigma) & = & 1+\mathsf{size}(\sigma) \end{array}$$

Proposition 20 (Complexity of $\mathbf{Decide}_{\dashv\|}$)**.**
The complexity of the algorithm $\mathbf{Decide}_{\dashv\|}$ *is* $\mathcal{O}(n^2)$*, where, for an argument* $\rho \curlywedge \sigma$*,* n *is the maximum between* $\mathsf{size}(\rho)$ *and* $\mathsf{size}(\sigma)$*.*

We remark that for obtaining such a complexity it is fundamental to avoid to recompute information, positive or negative, hence the need for sets A and F. It is also fundamental to be able to perform insertion and membership check in those sets in constant time. This can be done by implementing them as Boolean vectors indexed by all the pairs of subterms of the original ρ and σ.

From Proposition 20, using Theorem 12, it is immediate to get a decision procedure for the subcontract relation out of that for compliance:

$$\mathbf{Decide}_{\preccurlyeq_s}(\rho \preccurlyeq_s \sigma) = \mathbf{Decide}_{\dashv\|}\,(\overline{\rho} \curlywedge \sigma)$$

Corollary 21 (Complexity of $\mathbf{Decide}_{\preccurlyeq_s}$)**.**
The complexity of the algorithm $\mathbf{Decide}_{\preccurlyeq_s}$ *is* $\mathcal{O}(n^2)$*, where, for an argument* $\rho \preccurlyeq_s \sigma$*,* n *is the maximum between* $\mathsf{size}(\rho)$ *and* $\mathsf{size}(\sigma)$*.*

Proof. Immediate by noticing that building the dual of a given contract takes linear time. □

Remark 5. The polynomial decision procedure $\mathbf{Decide}_{\dashv\|}$ applies also to the formalism of retractable session contracts of [18]. In fact, the sound and complete formal system $\rhd_{\mathbf{rC}}$ (and the corresponding decision procedure) is the restriction to elements of $\mathbf{rC}$ of the system in Figure 4. Obviously, when applied to elements of $\mathbf{rC}$, the clauses -8- and -12- of $\mathbf{Decide}_{\dashv\|}$ do not need to take into account the possibility of internal input choices.

6. Retractable Contracts vs Reversible Computing

In this section we explore the relations between our retractable contracts and calculi for reversible computing (see [25] for an overview). In [16], Phillips and Ulidowski provide an automatic technique to derive, from the forward semantics of a given calculus, its reversible semantics. In principle, we would like to apply

the technique in [16] to the forward calculus underlying our retractable contracts, which is presented below. It is built by equipping retractable contracts with the semantics obtained by replacing both a and $\overline{a}$ with α in the semantics of session contracts (recalled in Definition 12 in Section 3.1.1).

Definition 19 (Retractable Contracts Underlying Semantics). The following rules define the LTS for retractable contracts and the reduction semantics for client/server pairs.

$$\alpha.\sigma \oplus \sigma' \xrightarrow{\tau}_U \alpha.\sigma \qquad \alpha.\sigma \xrightarrow{\alpha}_U \sigma \qquad \alpha.\sigma + \sigma' \xrightarrow{\alpha}_U \sigma$$

$$\frac{\rho \xrightarrow{\tau}_U \rho'}{\rho \parallel \sigma \longrightarrow_U \rho' \parallel \sigma} \qquad \frac{\sigma \xrightarrow{\tau}_U \sigma'}{\rho \parallel \sigma \longrightarrow_U \rho \parallel \sigma'} \qquad \frac{\rho \xrightarrow{\alpha}_U \rho' \qquad \sigma \xrightarrow{\overline{\alpha}}_U \sigma'}{\rho \parallel \sigma \longrightarrow_U \rho' \parallel \sigma'}$$

However, the direct application of the technique in [16] requires the forward semantics to be specified as an LTS and to satisfy a number of conditions. More precisely, the semantics should be specified by rules in the simple path format [16, Def. 2.1] which must be either static rules [16, Def. 2.2], choice rules [16, Def. 2.3] or choice axioms [16, Def. 2.4]. The semantics in Definition 19 is a two-level semantics, where the lower level is an LTS and the top level a reduction semantics. Furthermore, some of the rules do not satisfy the required conditions. For instance, the rule for internal choice does not belong to any of the classes of rules above.

Thus, in order to directly apply the technique, we transform the syntax and the semantics of our forward calculus as follows:

- we merge the two levels of syntax (contracts and client/server pairs) into one, and specify its semantics as an LTS by considering reductions as transitions with label τ;
- we transform internal choice into τ-prefixed external choice;
- we separate action prefixing from internal/external choice.

The resulting calculus is defined as follows.

Definition 20 (Transformed Contracts). The set TC of *transformed contracts* is the set of closed expressions generated by the following grammar.

$$\sigma := \alpha_\tau.\sigma \;\mid\; \sum_{i\in I} \sigma_i \;\mid\; x \;\mid\; \mathsf{rec}\, x.\sigma \;\mid\; \sigma \parallel \sigma' \;\mid\; \mathbf{1}$$

where α_τ denotes a, $\overline{a}$ or τ.

We define below a translation function $[\![\cdot]\!]$ from either a retractable contract σ or a client/server pair $\sigma \parallel \rho$ into transformed contracts.

Definition 21 (Translation Function). The translation function $[\![\cdot]\!] : \mathbf{rsCo} \cup (\mathbf{rsCo} \times \mathbf{rsCo}) \longrightarrow \mathsf{TC}$ is defined inductively as follows:

$$[\![\textstyle\sum_{i\in I} \alpha_i.\sigma_i]\!] = \sum_{i\in I} \alpha_i.[\![\sigma_i]\!] \qquad [\![x]\!] = x \qquad [\![\mathsf{rec}\, x.\sigma]\!] = \mathsf{rec}\, x.[\![\sigma]\!]$$
$$[\![\textstyle\bigoplus_{i\in I} \alpha_i.\sigma_i]\!] = \sum_{i\in I} \tau.\alpha_i.[\![\sigma_i]\!] \qquad [\![\mathbf{1}]\!] = \mathbf{1} \qquad [\![\sigma \parallel \rho]\!] = [\![\sigma]\!] \parallel [\![\rho]\!]$$

Transformed contracts allow for general parallel composition and mixed choice. However, the restriction of transformed contracts to the image of function $[\![\cdot]\!]$ is closed under reduction, as shown by the semantics below. Thus, from now on, we consider only the transformed contracts in the image of $[\![\cdot]\!]$.

Definition 22 (Semantics of Transformed Contracts). The rules below, plus their symmetric, define the semantics of transformed contracts.

$$\alpha_\tau.\sigma \xrightarrow{\alpha_\tau} \sigma \qquad \frac{\sigma \xrightarrow{\alpha_\tau} \sigma'}{\sigma + \rho \xrightarrow{\alpha_\tau} \sigma'}$$

$$\frac{\rho \xrightarrow{\tau} \rho'}{\rho \parallel \sigma \xrightarrow{\tau} \rho' \parallel \sigma} \qquad \frac{\rho \xrightarrow{\alpha} \rho' \qquad \sigma \xrightarrow{\overline{\alpha}} \sigma'}{\rho \parallel \sigma \xrightarrow{\tau} \rho' \parallel \sigma'}$$

It is easy to check that the LTS for transformed contracts and the LTS underlying retractable contracts model the same client/server interactions.

Proposition 22. *Let* $\sigma, \rho, \sigma', \rho' \in \mathbf{rs}\mathsf{C}$.

$$\sigma \parallel \rho \longrightarrow_U \sigma' \parallel \rho' \quad \textit{iff} \quad [\![\sigma \parallel \rho]\!] \xrightarrow{\tau} [\![\sigma' \parallel \rho']\!]$$

Proof. By inspection of the rules. □

One can apply to the LTS in Definition 22 the technique in [16], obtaining the LTS below. In order to simplify the treatment, we replaced the keys used in [16] to annotate actions with an underline. While this is not correct in general, it is correct in the image of our client/server pairs, since keys are used to distinguish interactions with different communication partners, but in our case for each action there is at most one possible partner.

Definition 23 (Reversible Transformed Contracts). The rules below, plus their symmetric, define the *forward semantics* of reversible transformed contracts.

$$\frac{\mathrm{std}(X)}{\alpha_\tau.X \xrightarrow{\alpha_\tau} \underline{\alpha_\tau}.X} \qquad \frac{X \xrightarrow{\alpha_\tau} X'}{\underline{\alpha_\tau}.X \xrightarrow{\alpha_\tau} \underline{\alpha_\tau}.X'} \qquad \frac{X \xrightarrow{\alpha_\tau} X' \quad \mathrm{std}(Y)}{X + Y \xrightarrow{\alpha_\tau} X' + Y}$$

$$\frac{X \xrightarrow{\tau} X'}{X \parallel Y \xrightarrow{\tau} X' \parallel Y} \qquad \frac{X \xrightarrow{\alpha} X' \qquad Y \xrightarrow{\overline{\alpha}} Y'}{X \parallel Y \xrightarrow{\tau} X' \parallel Y'}$$

In the rules, $\mathrm{std}(X)$ holds if X does not contain underlined prefixes.

The *backward semantics* of reversible transformed contracts, denoted by arrow $\rightsquigarrow$, is obtained by changing the direction of the arrows in the rules above.

Figure 12 shows how a sample reduction sequence for a retractable client/server pair corresponds to a reduction sequence for the corresponding transformed contract. We formalize the correspondence hinted at in the example by defining a notion of *barbed simulation* between retractable client/server pairs and transformed contracts, and showing that each client/server pair (with empty histories) is simulated by its translation (see Appendix A.4 for the proof). We start by defining barbs for both retractable client-server pairs and reversible transformed contracts.

$[\![a.c+b.d]\!] = a.c+b.d$			$[\![\overline{a}.\overline{c}+\overline{b}.(e\oplus f)]\!] = \overline{a}.\overline{c}+\overline{b}.(\tau.e+\tau.f)$		
$\langle\,\rangle \ltimes a.c+b.d$	$\parallel$	$\langle\,\rangle \ltimes \overline{a}.\overline{c}+\overline{b}.(e\oplus f)$	$a.c+b.d$	$\parallel$	$\overline{a}.\overline{c}+\overline{b}.(\tau.e+\tau.f)$
$\longrightarrow_f \quad a.c \ltimes d$	$\parallel$	$\overline{a}.\overline{c} \ltimes e\oplus f$	$\longrightarrow \quad a.c+\underline{b}.d$	$\parallel$	$\overline{a}.\overline{c}+\overline{b}.(\tau.e+\tau.f)$
$\longrightarrow_f \quad a.c \ltimes d$	$\parallel$	$\overline{a}.\overline{c} \ltimes e$	$\longrightarrow \quad a.c+\underline{b}.d$	$\parallel$	$\overline{a}.\overline{c}+\overline{b}.(\underline{\tau}.e+\tau.f)$
			$\rightsquigarrow \quad a.c+\underline{b}.d$	$\parallel$	$\overline{a}.\overline{c}+\overline{b}.(\tau.e+\tau.f)$
$\longrightarrow_b \quad \langle\,\rangle \ltimes a.c$	$\parallel$	$\langle\,\rangle \ltimes \overline{a}.\overline{c}$	$\rightsquigarrow \quad a.c+b.d$	$\parallel$	$\overline{a}.\overline{c}+\overline{b}.(\tau.e+\tau.f)$
$\longrightarrow_f \quad \circ \ltimes c$	$\parallel$	$\circ \ltimes \overline{c}$	$\longrightarrow \quad \underline{a}.c+b.d$	$\parallel$	$\underline{\overline{a}}.\overline{c}+\overline{b}.(\tau.e+\tau.f)$
$\longrightarrow_f \quad \circ\colon\circ \ltimes \mathbf{1}$	$\parallel$	$\circ\colon\circ \ltimes \mathbf{1}$	$\longrightarrow \quad \underline{a}.\underline{c}+b.d$	$\parallel$	$\underline{\overline{a}}.\underline{\overline{c}}+\overline{b}.(\tau.e+\tau.f)$

Figure 12: Retractable client/server pairs vs transformed contracts: an example

Definition 24 (Barbs).
Given a retractable client-server pair $\mathrm{H}_1 \ltimes \rho \parallel \mathrm{H}_2 \ltimes \sigma$ and a name or coname α, we say that $\mathrm{H}_1 \ltimes \rho \parallel \mathrm{H}_2 \ltimes \sigma$ has a barb at α, written $\mathrm{H}_1 \ltimes \rho \parallel \mathrm{H}_2 \ltimes \sigma \downarrow_\alpha$, when defined by the following structural induction on $\mathrm{H}_1 \ltimes \rho \parallel \mathrm{H}_2 \ltimes \sigma$:

$$\begin{array}{rcl}
\mathrm{H}_1 \ltimes \rho \parallel \mathrm{H}_2 \ltimes \sigma \downarrow_\alpha & \text{iff} & \mathrm{H}_1 \ltimes \rho \downarrow_\alpha \text{ or } \mathrm{H}_2 \ltimes \sigma \downarrow_\alpha \\
\mathrm{H}_1 \ltimes \rho \downarrow_\alpha & \text{iff} & \rho \downarrow_\alpha \\
\sum_{i\in I} \alpha_i.\sigma_i \downarrow_{\alpha_i} & & \\
\bigoplus_{i\in I} \alpha_i.\sigma_i \downarrow_{\alpha_i} & & \\
\mathsf{rec}\, x.\sigma \downarrow_\alpha & \text{iff} & \sigma \downarrow_\alpha
\end{array}$$

Similarly, given a reversible transformed contract X and a name or coname α, we say that X has a barb at α, written $X \downarrow_\alpha$, when defined by the following structural induction on X:

$$\begin{array}{rcl}
\alpha.X \downarrow_\alpha & & \\
\tau.X \downarrow_\alpha & \text{iff} & X \downarrow_\alpha \\
\underline{\beta}.X \downarrow_\alpha & \text{iff} & X \downarrow_\alpha \\
\underline{\tau}.X \downarrow_\alpha & \text{iff} & X \downarrow_\alpha \\
\sum_{i\in I} X_i \downarrow_\alpha & \text{iff} & (\mathrm{std}(\sum_{i\in I} X_i) \wedge \exists i.X_i \downarrow_\alpha) \text{ or } (\exists i.X_i = \underline{\alpha_\tau} \wedge X_i \downarrow_\alpha) \\
X \parallel X' \downarrow_\alpha & \text{iff} & X \downarrow_\alpha \text{ or } X' \downarrow_\alpha \\
\mathsf{rec}\, x.X \downarrow_\alpha & \text{iff} & X \downarrow_\alpha
\end{array}$$

While the definition of barbs for retractable client-server pairs is standard, the one for transformed contracts has a few peculiarities, that we now discuss. The main point is that underlined actions are actions that have already been executed, hence they do not show barbs, but propagate barbs from their continuation. This is particularly tricky in the case of choice: if the transformed contract is not standard (that is, it has some underlined action), then only the selected branch (that is, the one with an underlined action) needs to be inspected for barbs. For instance, $a.c+b.d \downarrow_a$ and $a.c+b.d \downarrow_b$, while $\underline{a}.c+b.d \downarrow_c$ and has no other barb. We also notice that τ actions propagate barbs from their continuation.

Definition 25 (Barbed simulation).

Let R be a relation between retractable client/server pairs and reversible transformed contracts. R is a barbed simulation iff for each $(\mathrm{H}_1 \ltimes \rho \parallel \mathrm{H}_2 \ltimes \sigma, X) \in R$:

- if $\mathrm{H}_1 \ltimes \rho \parallel \mathrm{H}_2 \ltimes \sigma \longrightarrow_f \mathrm{H}'_1 \ltimes \rho' \parallel \mathrm{H}'_2 \ltimes \sigma'$ then $X \xrightarrow{\tau} X'$ and $(\mathrm{H}'_1 \ltimes \rho' \parallel \mathrm{H}'_2 \ltimes \sigma', X') \in R$;
- if $\mathrm{H}_1 \ltimes \rho \parallel \mathrm{H}_2 \ltimes \sigma \longrightarrow_b \mathrm{H}'_1 \ltimes \rho' \parallel \mathrm{H}'_2 \ltimes \sigma'$ then $X \overset{\tau}{\rightsquigarrow}^{+} X'$ and $(\mathrm{H}'_1 \ltimes \rho' \parallel \mathrm{H}'_2 \ltimes \sigma', X') \in R$;
- if $\mathrm{H}_1 \ltimes \rho \parallel \mathrm{H}_2 \ltimes \sigma \downarrow_\alpha$ then $X \downarrow_\alpha$;

where $\overset{\tau}{\rightsquigarrow}^{+}$ is the transitive closure of $\overset{\tau}{\rightsquigarrow}$.

Theorem 23.
For any $\rho, \sigma \in \mathbf{rs}\mathsf{C}$ there is a simulation R such that $(\langle\,\rangle \ltimes \rho \parallel \langle\,\rangle \ltimes \sigma, [\![\rho \parallel \sigma]\!]) \in R$.

Note that the opposite of Theorem 23 cannot hold because of the mechanisms to control reversibility discussed in Remark 2. Indeed, the technique in [16] generates an uncontrolled semantics. A sample difference is that in transformed contracts we can have an infinite reduction sequence persistently choosing the right branch after the backward reduction, as follows.

$$\begin{array}{rlcll} & a.c + b.d \parallel \overline{a}.\overline{c} + \overline{b}.(\tau.e + \tau.f) & \longrightarrow^{+} & a.c + \underline{b}.d \parallel \overline{a}.\overline{c} + \overline{\underline{b}}.(\underline{\tau}.e + \tau.f) \\ \rightsquigarrow^{+} & a.c + b.d \parallel \overline{a}.\overline{c} + \overline{b}.(\tau.e + \tau.f) & \longrightarrow^{+} & a.c + \underline{b}.d \parallel \overline{a}.\overline{c} + \overline{\underline{b}}.(\underline{\tau}.e + \tau.f) \\ \rightsquigarrow^{+} & a.c + b.d \parallel \overline{a}.\overline{c} + \overline{b}.(\tau.e + \tau.f) & \longrightarrow^{+} & \text{etc.} \end{array}$$

It is easy to check, instead, that $\langle\,\rangle \ltimes a.c + b.d \parallel \langle\,\rangle \ltimes \overline{a}.\overline{c} + \overline{b}.(e \oplus f)$ can perform no infinite reduction sequence since the chosen branch is discarded upon rollback.

We remark that the present investigation just relates the retractable semantics of contracts and reversible calculi, showing that the retractable semantics correctly models a form of reversibility. In itself, this provides little justification to the speculative semantics. However, since the retractable and the speculative semantics give rise to the same notion of compliance, if the retractable semantics is correct than also the speculative semantics needs to be correct, at least from the abstract point of view determined by considering compliance.

7. Related Work and Conclusion

We have presented two conservative extensions of the session contracts of [32, 13, 14], a formalism interpreting session types [4] into a subset of contracts [1, 2, 3]. One extension deals with backtracking and one with speculative execution. We have shown that they both give rise to *the same* compliance relation, and, as a consequence, to the same subcontract (both for servers and for clients) and duality relations. For each of these relations we provided syntactic characterizations of the semantic concepts, allowing for efficient ways of checking them.

We discussed in the Introduction the improvements with respect to the previous work presented in [18, 17]. Another closely related work is [33, 34],

where a different form of contracts with rollback is presented. Our retractable contracts depart from that model on three main aspects: (1) we use rollback in a disciplined way to tolerate failures in the interaction, thus making it easier for contracts to be compliant, while in [33, 34] the decision to rollback is internal to each participant, and the partner needs to be compliant with both forward behavior and all the behaviors emerging from each possible rollback; (2) we embed checkpoints in the structure of contracts, avoiding explicit checkpoints; (3) we keep a stack of "pasts", instead of just a single past as in [33, 34].

Reversibility, generalizing backtracking by allowing one to go back to any past state, has also been studied in the setting of binary session types [35, 36, 37]. There however the emphasis is on defining the reversible engine, based on causal-consistent reversibility [25], and not on studying compliance or subtyping (which would correspond to our subcontract relation).

Similarly to our retractable contracts, long running transactions with compensations, and in particular interacting transactions [38], allow one to undo past agreements. In interacting transactions, however, abort (which corresponds to our backtracking) can occur at any time, not only when an agreement cannot be found as in our case. Also, each transaction offers just two possibilities, and they are sorted: first the normal execution, then the compensation. Finally, compliance of interacting transactions has never been studied.

In [39] a game-theoretical interpretation of the retractable session contracts of [18] has been provided. Such an interpretation is likely to extend to the retractable contracts presented here.

It would also be worth to investigate whether our approach can be extended to multi-party sessions [40]. Uncontrolled reversibility (according to the terminology in [25]) in multi-party sessions has been studied in [41]. An investigation of multi-party sessions with rollbacks and named checkpoints has also been already undertaken, in [42, 43]. In such a paper, however, the cause of a rollback is not a synchronization failure, but it is completely transparent to the calculus. Moreover, chosen branches are not discarded and can be retried upon rollback.

Because of the relevance of higher-order features in type systems, and of session delegation in type systems with sessions in particular, also higher-order session contracts, i.e. session contracts with delegation, have been investigated [13, 44]. It is hence worth studying the integration of backtracking (or speculative execution) and session delegation.

As mentioned in Example 2, any possible preference among alternatives in external output choices is abstracted away in our contracts. Actually, it would be possible to model preference by dropping commutativity of $+$ and thus considering an order for alternatives in choices, with the highest priority option selected always first. Such a possibility has been explored in [26], in the context of orchestrated session types. There the information obtained when compliance is decided is embodied in an orchestrator enabling to drive the choices of processes. In presence of a choice which can possibly lead to a failure, the orchestrator forces the processes to follow one of the *safe* branches, so avoiding the need for mechanisms such as backtracking or speculative execution. Instead, a setting combining rollback and preference among alternatives has been studied from an

operational point of view in [45].

A last line of future work concerns the applicability of our results to real world applications. In general, contracts can be extracted from the specification or the code of clients and servers (see for instance [46], where a form of contracts is extracted from Java programs and exploited for deadlock analysis), and then used to check compliance, thus guaranteeing successful interaction. Of course, our contracts come handy when clients and servers use either backtracking, as provided, e.g., by the CBack library [47] for C, or speculative parallelism, as provided, e.g., by the C# library described in [8]. How to actually extract from client/server software based on such or similar libraries the corresponding retractable or speculative contracts in a rigorous and possibly automatic way is still to be seen.

Acknowledgments

This work was partially supported by the COST Action IC1405 on "Reversible computation - extending horizons of computing". The first and third authors were partially supported also by the COST Action EUTYPES CA-15123 and, respectively, Project "Chance" of the University of Catania and Project FORMS 2015 of Turin. The second author is member of the INdAM Research group GNCS. We thank Mariangiola Dezani-Ciancaglini for interesting discussions and helpful suggestions. We also thank the anonymous reviewers for their useful comments and remarks.

References

[1] S. Carpineti, G. Castagna, C. Laneve, L. Padovani, A formal account of contracts for Web Services, in: WS-FM, no. 4184 in LNCS, Springer, 2006, pp. 148–162.

[2] C. Laneve, L. Padovani, The must preorder revisited: An algebraic theory for web services contracts, in: CONCUR, Vol. 4703 of LNCS, Springer, 2007, pp. 212–225.

[3] G. Castagna, N. Gesbert, L. Padovani, A theory of contracts for web services, ACM Trans. on Prog. Lang. and Sys. 31 (5) (2009) 19:1–19:61.

[4] K. Honda, V. T. Vasconcelos, M. Kubo, Language primitives and type disciplines for structured communication-based programming, in: ESOP, Vol. 1381 of LNCS, Springer, 1998, pp. 22–138.

[5] H. Hüttel, I. Lanese, V. T. Vasconcelos, L. Caires, M. Carbone, P.-M. Deniélou, D. Mostrous, L. Padovani, A. Ravara, E. Tuosto, H. T. Vieira, G. Zavattaro, Foundations of session types and behavioural contracts, ACM Comput. Surv. 49 (1) (2016) 3:1–3:36.

[6] M. Carbone, K. Honda, N. Yoshida, Structured interactional exceptions in session types, in: CONCUR, Vol. 5201 of LNCS, Springer, 2008, pp. 402–417.

[7] M. Bartoletti, T. Cimoli, M. Murgia, A. Podda, L. Pompianu, Compliance and subtyping in timed session types, in: FORTE, Vol. 9039 of LNCS, Springer, 2015, pp. 161–177.

[8] P. Prabhu, G. Ramalingam, K. Vaswani, Safe programmable speculative parallelism, in: PLDI, ACM, 2010, pp. 50–61.

[9] A. Avizienis, J. Laprie, B. Randell, C. E. Landwehr, Basic concepts and taxonomy of dependable and secure computing, IEEE Trans. Dep. Sec. Comput. 1 (1) (2004) 11–33.

[10] C. D. Carothers, K. S. Perumalla, R. Fujimoto, Efficient optimistic parallel simulations using reverse computation, ACM Trans. Model. Comput. Simul. 9 (3) (1999) 224–253.

[11] C. G. Quiñones, C. Madriles, J. Sánchez, P. Marcuello, A. González, D. M. Tullsen, Mitosis compiler: An infrastructure for speculative threading based on pre-computation slices, in: PLDI, ACM, 2005, pp. 269–279.

[12] M. Dalla Preda, M. Gabbrielli, I. Lanese, J. Mauro, G. Zavattaro, Graceful interruption of request-response service interactions, in: ICSOC, Vol. 7084 of LNCS, Springer, 2011, pp. 590–600.

[13] F. Barbanera, U. de'Liguoro, Sub-behaviour relations for session-based client/server systems, MSCS 25 (6) (2015) 1339–1381.

[14] G. T. Bernardi, M. Hennessy, Modelling session types using contracts, Mathematical Structures in Computer Science 26 (3) (2016) 510–560.

[15] C. A. R. Hoare, Communicating Sequential Processes, Prentice-Hall, 1985.

[16] I. C. C. Phillips, I. Ulidowski, Reversing algebraic process calculi, J. of Logic and Alg. Progr. 73 (1-2) (2007) 70–96.

[17] F. Barbanera, I. Lanese, U. de'Liguoro, Retractable and speculative contracts, in: COORDINATION, Vol. 10319 of LNCS, Springer, 2017, pp. 119–137.

[18] F. Barbanera, M. Dezani-Ciancaglini, I. Lanese, U. de'Liguoro, Retractable contracts, in: PLACES 2015, Vol. 203 of EPTCS, Open Publishing Association, 2016, pp. 61–72.

[19] J. C. M. Baeten, J. A. Bergstra, J. W. Klop, W. P. Weijland, Term-rewriting systems with rule priorities, Theor. Comput. Sci. 67 (2&3) (1989) 283–301.

[20] A. Laville, Comparison of priority rules in pattern matching and term rewriting, J. Symbolic Computation 11 (1991) 321–347.

[21] M. R. Mousavi, I. C. C. Phillips, M. A. Reniers, I. Ulidowski, The meaning of ordered SOS, in: FSTTCS, Vol. 4337 of LNCS, Springer, 2006, pp. 333–344.

[22] M. R. Mousavi, I. Phillips, M. A. Reniers, I. Ulidowski, Semantics and expressiveness of ordered SOS, Inf. Comput. 207 (2) (2009) 85–119.

[23] V. Danos, J. Krivine, Reversible communicating systems, in: CONCUR, Vol. 3170 of LNCS, Springer, 2004, pp. 292–307.

[24] I. Lanese, C. A. Mezzina, J. Stefani, Reversibility in the higher-order π-calculus, Theor. Comput. Sci. 625 (2016) 25–84.

[25] I. Lanese, C. A. Mezzina, F. Tiezzi, Causal-consistent reversibility, Bulletin of the EATCS 114.

[26] F. Barbanera, U. de'Liguoro, Session types for orchestrated interactions, in: ICE@DisCoTec, Vol. 261 of EPTCS, 2017, pp. 17–36.

[27] G. Boudol, I. Castellani, M. Hennessy, A. Kiehn, Observing localities, Theor. Comput. Sci. 114 (1) (1993) 31–61.

[28] L. Zhang, S. Malik, The quest for efficient boolean satisfiability solvers, in: CAV, Vol. 2404 of LNCS, Springer, 2002, pp. 17–36.

[29] L. Xu, F. Hutter, H. H. Hoos, K. Leyton-Brown, Satzilla: Portfolio-based algorithm selection for SAT, J. Artif. Intell. Res. 32 (2008) 565–606.

[30] L. Padovani, Contract-based discovery of web services modulo simple orchestrators, Theoretical Computer Science 411 (2010) 3328–3347.

[31] V. Gapeyev, M. Y. Levin, B. C. Pierce, Recursive subtyping revealed, J. Funct. Program. 12 (6) (2002) 511–548.

[32] F. Barbanera, U. de'Liguoro, Two notions of sub-behaviour for session-based client/server systems, in: PPDP, ACM Press, 2010, pp. 155–164.

[33] F. Barbanera, M. Dezani-Ciancaglini, U. de'Liguoro, Compliance for reversible client/server interactions, in: BEAT, Vol. 162 of EPTCS, 2014, pp. 35–42.

[34] F. Barbanera, M. Dezani-Ciancaglini, U. de'Liguoro, Reversible client/server interactions, Formal Asp. Comput. 28 (4) (2016) 697–722.

[35] F. Tiezzi, N. Yoshida, Towards reversible sessions, in: PLACES, Vol. 155 of EPTCS, 2014, pp. 17–24.

[36] F. Tiezzi, N. Yoshida, Reversible session-based pi-calculus, J. Log. Algebr. Meth. Program. 84 (5) (2015) 684–707.

[37] C. A. Mezzina, J. A. Pérez, Reversibility in session-based concurrency: A fresh look, J. Log. Algebr. Meth. Program. 90 (2017) 2–30.

[38] E. de Vries, V. Koutavas, M. Hennessy, Communicating transactions - (extended abstract), in: CONCUR, Vol. 6269 of LNCS, Springer, 2010, pp. 569–583.

[39] F. Barbanera, U. de'Liguoro, Retractability, games and orchestrators for session contracts, Logical Methods in Computer Science Volume 13, Issue 3.

[40] K. Honda, N. Yoshida, M. Carbone, Multiparty asynchronous session types, in: POPL, ACM Press, 2008, pp. 273–284.

[41] C. A. Mezzina, J. A. Pérez, Causally consistent reversible choreographies: A monitors-as-memories approach, in: PPDP, ACM, 2017, pp. 127–138.

[42] M. Dezani-Ciancaglini, P. Giannini, Reversible multiparty sessions with checkpoints, in: EXPRESS/SOS, Vol. 222 of EPTCS, 2016, pp. 60–74.

[43] I. Castellani, M. Dezani-Ciancaglini, P. Giannini, Concurrent reversible sessions, in: CONCUR, Vol. 85 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017, pp. 30:1–30:17.

[44] G. T. Bernardi, M. Hennessy, Using higher-order contracts to model session types, Logical Methods in Computer Science 12 (2).

[45] I. Lanese, M. Lienhardt, C. A. Mezzina, A. Schmitt, J. Stefani, Concurrent flexible reversibility, in: ESOP, Vol. 7792 of LNCS, Springer, 2013, pp. 370–390.

[46] A. Garcia, C. Laneve, Deadlock detection of java bytecode, CoRR abs/1709.04152. arXiv:1709.04152.
URL `http://arxiv.org/abs/1709.04152`

[47] K. Helsgaun, Cback: A simple tool for backtrack programming in C, Softw., Pract. Exper. 25 (8) (1995) 905–934.

Appendix A. Proofs

Appendix A.1. Proofs of Section 3

Proof of Theorem 1
It is straightforward to check the following

Fact 24. *i)* $\mathbf{Prove}(\Gamma \rhd \rho \backsim \sigma) \neq \mathbf{fail}$ *iff* $\Gamma \rhd \rho \backsim \sigma$.

ii) $\mathbf{Prove}(\Gamma \rhd \rho \backsim \sigma) = \mathcal{D} \neq \mathbf{fail}$ *implies* $\mathcal{D}$ *has conclusion* $\Gamma \rhd \rho \backsim \sigma$.

By Fact 24, we only need to show that the procedure **Prove** always terminates. Note 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, respectively, ρ and σ (because of the equi-recursive view of recursion they can also be ρ and σ). Since contract expressions generate regular trees, there are only finitely many such subexpressions. This implies that the number of different calls of procedure **Prove** is always finite. □

Soundness and Completeness Proofs (Theorems 2 and 3)
We begin with the proof of Soundness and Completeness of system $\rhd$ with respect to the retractable compliance.

Retractable Soundness and Completeness. If a configuration is stuck, then both histories are empty. This is a consequence of the fact that the property "the histories of client and server have the same length" is preserved by reductions.

Lemma 25. *If* $\langle\,\rangle \ltimes \rho \parallel \langle\,\rangle \ltimes \sigma \xrightarrow{*} \mathrm{H}_1 \ltimes \rho' \parallel \mathrm{H}_2 \ltimes \sigma' \not\longrightarrow$, *then* $\mathrm{H}_1 = \mathrm{H}_2 = \langle\,\rangle$.

Proof. Clearly $\mathrm{H}_1 \ltimes \rho' \parallel \mathrm{H}_2 \ltimes \sigma' \not\longrightarrow$ implies either $\mathrm{H}_1 = \langle\,\rangle$ or $\mathrm{H}_2 = \langle\,\rangle$. Observe that: rule (comm) adds one element to both stacks; rule (τ) does not modify any of the stacks; rule (rbk) removes one element from both stacks.
Then starting from two stacks containing the same number of elements, the reduction always produces two stacks containing the same number of elements. So $\mathrm{H}_1 = \langle\,\rangle$ implies $\mathrm{H}_2 = \langle\,\rangle$ and vice versa. □

The following lemma proves that compliance is preserved by the concatenation of histories to the left of the current histories.

Lemma 26. *If* $\mathrm{H}_1 \ltimes \rho \dashv\!\!\parallel^{\mathsf{R}} \mathrm{H}_2 \ltimes \sigma$, *then* $\mathrm{H}'_1 : \mathrm{H}_1 \ltimes \rho \dashv\!\!\parallel^{\mathsf{R}} \mathrm{H}'_2 : \mathrm{H}_2 \ltimes \sigma$ *for all* H'_1, H'_2.

Proof sketch. It suffices to show that if in a sequence of reductions H'_1 or H'_2 are used, then $\mathrm{H}_1 \ltimes \rho \not\dashv\!\!\parallel^{\mathsf{R}} \mathrm{H}_2 \ltimes \sigma$. □

The following lemma gives all possible shapes of compliant contracts.

Lemma 27. *We have* $\rho \dashv\!\!\parallel^{\mathsf{R}} \sigma$ *iff 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* $\exists k \in I \cap J.\ \rho_k \dashv\!\!\parallel^{\mathsf{R}} \sigma_k$;
3. $\rho = \bigoplus_{i\in I} \overline{\alpha}_i.\rho_i$, $\sigma = \sum_{j\in J} \alpha_j.\sigma_j$, $I \subseteq J$ *and* $\forall k \in I.\ \rho_k \dashv\!\!\parallel^{\mathsf{R}} \sigma_k$;
4. $\rho = \sum_{i\in I} \overline{\alpha}_i.\rho_i$, $\sigma = \bigoplus_{j\in J} \alpha_j.\sigma_j$, $I \supseteq J$ *and* $\forall k \in J.\ \rho_k \dashv\!\!\parallel^{\mathsf{R}} \sigma_k$.

Proof. The if part is immediate. We prove the only if part by contraposition and by cases on the possible shapes of ρ and σ.
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 $\rho_{k_i} \not\dashv\!\!\parallel^{\mathsf{R}} \sigma_{k_i}$ for $1 \le i \le n$. Then we get

$$\langle\,\rangle \ltimes \rho_{k_i} \parallel \langle\,\rangle \ltimes \sigma_{k_i} \xrightarrow{*} \mathrm{H}_i \ltimes \rho'_i \parallel \mathrm{H}'_i \ltimes \sigma'_i \not\longrightarrow$$

for $1 \le i \le n$, where $\rho'_i \neq \mathbf{1}$ and $\mathrm{H}_i = \mathrm{H}'_i = \langle\,\rangle$ by Lemma 25. This implies

$$\begin{array}{rl} & \sum_{i\in I\setminus\{k_1\}} \alpha_i.\rho_i \ltimes \rho_{k_1} \parallel \sum_{j\in J\setminus\{k_1\}} \overline{\alpha}_j.\sigma_j \ltimes \sigma_{k_1} \\ \xrightarrow{*} & \sum_{i\in I\setminus\{k_1\}} \alpha_i.\rho_i \ltimes \rho'_1 \parallel \sum_{j\in J\setminus\{k_1\}} \overline{\alpha}_j.\sigma_j \ltimes \sigma'_1 \end{array}$$

by Lemma 26. Let $I' = I \setminus J$ and $J' = J \setminus I$. We can reduce $\langle\,\rangle \ltimes \rho \parallel \langle\,\rangle \ltimes \sigma$ only as follows:

$$\begin{array}{rlll} \langle\,\rangle \ltimes \rho \parallel \langle\,\rangle \ltimes \sigma & \longrightarrow & \sum_{i\in I\setminus\{k_1\}} \alpha_i.\rho_i \ltimes \rho_{k_1} \parallel \sum_{j\in J\setminus\{k_1\}} \overline{\alpha}_j.\sigma_j \ltimes \sigma_{k_1} & \text{by (comm)} \\ & \xrightarrow{*} & \sum_{i\in I\setminus\{k_1\}} \alpha_i.\rho_i \ltimes \rho'_1 \parallel \sum_{j\in J\setminus\{k_1\}} \overline{\alpha}_j.\sigma_j \ltimes \sigma'_1 & \\ & \longrightarrow & \langle\,\rangle \ltimes \sum_{i\in I\setminus\{k_1\}} \alpha_i.\rho_i \parallel \langle\,\rangle \ltimes \sum_{j\in J\setminus\{k_1\}} \overline{\alpha}_j.\sigma_j & \text{by (rbk)} \\ & \vdots & \vdots & \\ & \xrightarrow{*} & \sum_{i\in I'} \alpha_i.\rho_i \ltimes \rho'_n \parallel \sum_{j\in J'} \overline{\alpha}_j.\sigma_j \ltimes \sigma'_n & \\ & \longrightarrow & \langle\,\rangle \ltimes \sum_{i\in I'} \alpha_i.\rho_i \parallel \langle\,\rangle \ltimes \sum_{j\in J'} \overline{\alpha}_j.\sigma_j & \text{by (rbk)} \end{array}$$

and $\langle\,\rangle \ltimes \sum_{i\in I'} \alpha_i.\rho_i \parallel \langle\,\rangle \ltimes \sum_{j\in J'} \overline{\alpha}_j.\sigma_j$ 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

$$\langle\,\rangle \ltimes \rho \parallel \langle\,\rangle \ltimes \sigma \longrightarrow \langle\,\rangle \ltimes \overline{\alpha}_k.\rho_k \parallel \langle\,\rangle \ltimes \sigma \not\longrightarrow$$

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

$$\langle\,\rangle \ltimes \rho_k \parallel \langle\,\rangle \ltimes \sigma_k \xrightarrow{*} \langle\,\rangle \ltimes \rho' \parallel \langle\,\rangle \ltimes \sigma' \not\longrightarrow$$

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

which imply

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

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

Lemma 27 suggests that $\dashv\!\!\mid^{\mathsf{R}}$ can be coinductively defined, or equivalently that $\dashv\!\!\mid^{\mathsf{R}} = \bigcap_n \dashv\!\!\mid^{\mathsf{R}}_n$ where $\dashv\!\!\mid^{\mathsf{R}}_0$ is the trivial relation $\mathbf{rsC} \times \mathbf{rsC}$, and for all $n > 0$, $\mathbf{1} \dashv\!\!\mid^{\mathsf{R}}_n \sigma$ and we have $\rho \dashv\!\!\mid^{\mathsf{R}}_n \sigma$ if one of the following holds:

1. $\rho = \sum_{i\in I} \alpha_i.\rho_i$, $\sigma = \sum_{j\in J} \overline{\alpha}_j.\sigma_j$ and $\exists k \in I \cap J.\ \rho_k \dashv\!\!\mid^{\mathsf{R}}_{n-1} \sigma_k$;
2. $\rho = \bigoplus_{i\in I} \overline{\alpha}_i.\rho_i$, $\sigma = \sum_{j\in J} \alpha_j.\sigma_j$, $I \subseteq J$ and $\forall k \in I.\ \rho_k \dashv\!\!\mid^{\mathsf{R}}_{n-1} \sigma_k$;
3. $\rho = \sum_{i\in I} \overline{\alpha}_i.\rho_i$, $\sigma = \bigoplus_{j\in J} \alpha_j.\sigma_j$, $I \supseteq J$ and $\forall k \in J.\ \rho_k \dashv\!\!\mid^{\mathsf{R}}_{n-1} \sigma_k$.

We write:

1. $\models^{\mathsf{R}} \Gamma$ if for all $\rho' \dashv\!\!\mid^{\mathsf{R}} \sigma' \in \Gamma$ we have $\rho' \dashv\!\!\mid^{\mathsf{R}} \sigma'$
2. $\Gamma \models^{\mathsf{R}} \rho \rightharpoonup\!\!\dashv \sigma$ if $\models^{\mathsf{R}} \Gamma$ implies $\rho \dashv\!\!\mid^{\mathsf{R}} \sigma$

We also write $\Gamma \models^{\mathsf{R}}_n \rho \rightharpoonup\!\!\dashv \sigma$ if $\dashv\!\!\mid^{\mathsf{R}}$ is replaced by $\dashv\!\!\mid^{\mathsf{R}}_n$ in the above.

Observing that $\dashv\!\!\mid^{\mathsf{R}}_{n+1} \subseteq \dashv\!\!\mid^{\mathsf{R}}_n$, we have that $\models^{\mathsf{R}}_{n+1} \Gamma$ implies $\models^{\mathsf{R}}_n \Gamma$. Also it is immediate to verify that the following holds:

Fact 28. *If $\Gamma \models^{\mathsf{R}}_n \rho \rightharpoonup\!\!\dashv \sigma$ for all n, then $\Gamma \models^{\mathsf{R}} \rho \rightharpoonup\!\!\dashv \sigma$.*

We can now prove Theorem 2, which can be restated as follows.

Theorem 2 (Retractable Soundness and Completeness)

$$\rhd \rho \rightharpoonup\!\!\dashv \sigma \quad \text{iff} \quad \models^{\mathsf{R}} \rho \rightharpoonup\!\!\dashv \sigma$$

Proof. ($\Rightarrow$) For this direction we can actually prove a stronger statement, namely $\quad \Gamma \rhd \rho \backsim \sigma \;\Rightarrow\; \Gamma \models^{\mathsf{R}} \rho \backsim \sigma$
By Fact 28 it suffices to prove that if $\Gamma \rhd \rho \backsim \sigma$ then $\Gamma \models^{\mathsf{R}}_n \rho \backsim \sigma$ for all n, which we establish by simultaneous induction over n and over the derivation $\mathcal{D}$ of $\Gamma \rhd \rho \backsim \sigma$.

If $\mathcal{D}$ ends by either Ax or Hyp then the thesis trivially holds. If $\mathcal{D}$ ends by:

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

then we have to show that $\models^{\mathsf{R}}_n \Gamma$ implies $\alpha.\rho + \rho' \dashv\!|\!|^{\mathsf{R}}_n \;\overline{\alpha}.\sigma + \sigma'$. By induction over n we know that $\Gamma \models^{\mathsf{R}}_{n-1} \alpha.\rho + \rho' \backsim \overline{\alpha}.\sigma + \sigma'$; from this and the fact that $\models^{\mathsf{R}}_n \Gamma$ implies $\models^{\mathsf{R}}_{n-1} \Gamma$, we obtain that $\alpha.\rho + \rho' \dashv\!|\!|^{\mathsf{R}}_{n-1} \overline{\alpha}.\sigma + \sigma'$, and hence that $\models^{\mathsf{R}}_{n-1} \Gamma, (\alpha.\rho + \rho' \backsim \overline{\alpha}.\sigma + \sigma')$. By induction over $\mathcal{D}$ it follows that $\rho \dashv\!|\!|^{\mathsf{R}}_{n-1} \sigma$, which implies $\alpha.\rho + \rho' \dashv\!|\!|^{\mathsf{R}}_n \overline{\alpha}.\sigma + \sigma'$ by definition of $\dashv\!|\!|^{\mathsf{R}}_n$, as desired.
The cases in which $\mathcal{D}$ ends by either $(\oplus \cdot +)$ or $(+ \cdot \oplus)$ are similar.
($\Leftarrow$) By Theorem 1 each computation of $\mathbf{Prove}(\rhd\; \rho \backsim \sigma)$ always terminates. By Lemma 27 and Fact 24, $\rho \dashv\!|\!|^{\mathsf{R}} \sigma$ implies that $\mathbf{Prove}(\rhd\; \rho \backsim \sigma) \neq \mathbf{fail}$, and hence $\rhd\; \rho \backsim \sigma$. □

We proceed now with the proof of Soundness and Completeness of system $\rhd$ with respect to the speculative compliance.

Speculative Soundness and Completeness. A lemma similar to Lemma 27 holds for speculative compliance as well.

Lemma 29. *We have $\rho \dashv\!|\!|^{\mathsf{S}} \sigma$ 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* $\exists k \in I \cap J.\ \rho_k \dashv\!|\!|^{\mathsf{S}} \sigma_k$;
3. $\rho = \bigoplus_{i\in I} \overline{\alpha}_i.\rho_i$, $\sigma = \sum_{j\in J} \alpha_j.\sigma_j$, $I \subseteq J$ *and* $\forall k \in I.\ \rho_k \dashv\!|\!|^{\mathsf{S}} \sigma_k$;
4. $\rho = \sum_{i\in I} \overline{\alpha}_i.\rho_i$, $\sigma = \bigoplus_{j\in J} \alpha_j.\sigma_j$, $I \supseteq J$ *and* $\forall k \in J.\ \rho_k \dashv\!|\!|^{\mathsf{S}} \sigma_k$.

Proof. The if part is immediate. We prove the only if part by contraposition and by cases on the possible shapes of ρ and σ.
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 $\rho_{k_i} \not\dashv\!|\!|^{\mathsf{S}} \sigma_{k_i}$ for $1 \leq i \leq n$. Then we get $\quad \rho_{k_i} \parallel \sigma_{k_i} \xrightarrow{*} \mathbf{C}^{\rho}_i \parallel \mathbf{C}^{\sigma}_i \not\longrightarrow$
for $1 \leq i \leq n$, where $\mathbf{C}^{\rho}_i \neq \mathbf{C} \mid \alpha_1 @ \ldots @ \alpha_n @ \mathbf{1}$.

This implies

$$\begin{array}{l} \sum_{i\in I} \alpha_i.\rho_i \parallel \sum_{j\in J} \overline{\alpha}_j.\sigma_j \\ \xrightarrow{*} (\prod_{i\in I\cap J} \alpha_i @ \rho_i) \mid \sum_{i\in I\setminus J} \alpha_i.\rho_i \parallel (\prod_{j\in I\cap J} \overline{\alpha}_j @ \sigma_j) \mid \sum_{j\in J\setminus I} \overline{\alpha}_j.\sigma_j \end{array}$$

where $\prod_{i\in I} \mathbf{C}_i$ denotes the parallel composition of $\mathbf{C}_i$ for each $i \in I$.
Terms $\sum_{i\in I\setminus J} \alpha_i.\rho_i$ and $\sum_{j\in J\setminus I} \overline{\alpha}_j.\sigma_j$ cannot interact with other terms. Instead,

term $\prod_{i\in I\cap J}\alpha_i \mathbin{@} \rho_i$ can interact only with term $\prod_{j\in I\cap J}\overline{\alpha}_j \mathbin{@} \sigma_j$ and vice versa. The interaction follows the computations $\rho_{k_i} \parallel \sigma_{k_i} \stackrel{*}{\longrightarrow} \mathbf{C}_i^\rho \parallel \mathbf{C}_i^\sigma \not\longrightarrow$, with the added prefixes α_i and $\overline{\alpha}_i$. However, none of these computations produces a thread of the form $\alpha_1' \mathbin{@} \ldots \mathbin{@} \alpha_n' \mathbin{@} \mathbf{1}$, hence $\rho \not\dashv\vdash^{\mathsf{S}} \sigma$.
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

$$\rho \parallel \sigma \longrightarrow \overline{\alpha}_k.\rho_k \parallel \sigma \not\longrightarrow$$

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

$$\rho_k \parallel \sigma_k \stackrel{*}{\longrightarrow} \mathbf{C}^\rho \parallel \mathbf{C}^\sigma \not\longrightarrow \qquad \text{and}$$

$$\begin{array}{ll} & \rho \parallel \sigma \\ \longrightarrow & \overline{\alpha}_k.\rho_k \parallel \sigma \\ \longrightarrow & \overline{\alpha}_k \mathbin{@} \rho_k \parallel \alpha_k \mathbin{@} \sigma_k \mid \sum_{j\in J\setminus\{k\}} \alpha_j.\sigma_j \\ \stackrel{*}{\longrightarrow} & \overline{\alpha}_k \mathbin{@} \mathbf{C}^\rho \parallel \alpha_k \mathbin{@} \mathbf{C}^\sigma \mid \sum_{j\in J\setminus\{k\}} \alpha_j.\sigma_j \\ \not\longrightarrow & \end{array}$$

where $\alpha \mathbin{@} \mathbf{C}$ denotes $\prod_{i\in I} \alpha \mathbin{@} \mathbf{T}_i$ if $\mathbf{C} = \prod_{i\in I} \mathbf{T}_i$.
In both cases we conclude that $\rho \not\dashv\vdash^{\mathsf{S}} \sigma$.

The proof for the case $\rho = \sum_{i\in I}\overline{\alpha}_i.\rho_i$, $\sigma = \bigoplus_{j\in J}\alpha_j.\sigma_j$ is similar. □

As with $\dashv\vdash^{\mathsf{R}}$, we define the family of relations $\dashv\vdash^{\mathsf{S}}_n$ on the basis of Lemma 29, which are such that $\dashv\vdash^{\mathsf{S}} = \bigcap_n \dashv\vdash^{\mathsf{S}}_n$; similarly we define the respective notions $\Gamma \models^{\mathsf{S}} \rho \backsim \sigma$ and $\Gamma \models^{\mathsf{S}_n} \rho \backsim \sigma$. Speculative soundness and completeness can hence be restated as follows.

Theorem 3 (Speculative Soundness and Completeness)

$$\rhd \rho \backsim \sigma \;\text{ iff }\; \models^{\mathsf{S}} \rho \backsim \sigma$$

Proof. (⇒) This implication can be proved in the same way as Theorem 2.
(⇐) By Theorem 1 each computation of $\mathbf{Prove}(\rhd \rho \backsim \sigma)$ always terminates. By Lemma 27 and Fact 24, $\rho \dashv\vdash^{\mathsf{R}} \sigma$ implies that $\mathbf{Prove}(\rhd \rho \backsim \sigma) \neq \mathbf{fail}$, and hence $\rhd \rho \backsim \sigma$. □

Appendix A.1.1. Proofs of Section 3.1

Before proving Proposition 7 of Section 3.1, we need a simple technical lemma and a fact.

Lemma 30. *Let $\rho \in \mathsf{SC}$. Then either $\rho = \overline{a}.\rho_1 \oplus \rho_2$, or $\rho = \alpha.\rho'$, or $\rho = a.\rho_1 + \rho_2$, or $\rho = \mathbf{1}$. Moreover, for any* H,

i) $\overline{a}.\rho_1 \oplus \rho_2 \stackrel{\tau}{\longrightarrow} \overline{a}.\rho_1$ *if and only if* $\mathrm{H}\ltimes\overline{a}.\rho_1 \oplus \rho_2 \stackrel{\tau}{\longrightarrow} \mathrm{H}\ltimes\overline{a}.\rho_1$;

ii) $\alpha.\rho' \stackrel{\alpha}{\longrightarrow} \rho'$ *if and only if* $\mathrm{H}\ltimes\alpha.\rho' \stackrel{\alpha}{\longrightarrow} \mathrm{H}{:}\circ \ltimes \rho'$;

iii) $a.\rho_1 + \rho_2 \stackrel{a}{\longrightarrow} \rho_1$ *if and only if* $\mathrm{H}\ltimes a.\rho_1 + \rho_2 \stackrel{a}{\longrightarrow} \mathrm{H} : \rho_2 \ltimes \rho_1$

iv) $\rho = \mathbf{1}$ *if and only if* ($\mathrm{H} \ltimes \rho \not\xrightarrow{\mathcal{O}}$ *and* $\mathrm{H} \ltimes \rho \not\xrightarrow{\mathcal{I}}$).

Proof. Easy, by definition of session contract and by Definitions 4 and 23. □

Fact 31. *Let* $\rho, \sigma \in \mathsf{SC}$. $\mathrm{H}_1 \ltimes \rho \parallel \mathrm{H}_2 \ltimes \sigma \longrightarrow_f \mathrm{H}_1' \ltimes \rho' \parallel \mathrm{H}_2' \ltimes \sigma'$ *implies* $\rho', \sigma' \in \mathsf{SC}$

We can now proceed with the proof of Propositions 7.

Proof of Proposition 7
(i) The inclusion $\mathsf{SC} \subsetneq \mathbf{rs}\mathsf{C}$ holds by definition. Hence, given $\rho, \sigma \in \mathsf{SC}$ we have $\rho, \sigma \in \mathbf{rs}\mathsf{C}$ and $\mathrm{H}_1 \ltimes \rho, \mathrm{H}_2 \ltimes \sigma \in \mathbf{rs}\mathsf{CH}$.

(⇒) By induction on the length of the reduction sequence $\xrightarrow{*}_{\mathsf{SC}}$, using Definitions 4 and 12 and Lemma 30 to check all the possible cases for the reductions of client/server pairs.

(⇐) Using Fact 31 we first show that no pair of the form $\mathrm{H}_1 \ltimes \alpha.\rho_1 + \rho_2 \parallel \mathrm{H}_2 \ltimes \overline{\alpha}.\sigma_1 + \sigma_2$ can ever appear inside the reduction sequence $\xrightarrow{*}_f$. Then we proceed by induction on the length of the reduction sequence $\xrightarrow{*}_f$ using Definitions 4 and 12 and Lemma 30 to check all the possible cases for the reductions of client/server pairs with histories.

(ii) By induction on the length of the derivation. The base case is trivial. Let us consider the inductive case.

(⇒) the actions performed by σ' and ρ' can be performed as well by $\alpha_1 @ \ldots @ \alpha_n @ \rho'$ and $\overline{\alpha_1} @ \ldots \overline{\alpha_n} @ \sigma'$, with the only possible side effects of adding complementary actions to the prefixes, and of spawning further parallel threads. Since both the effects are compatible with the thesis, we are done.

(⇐) if the action is a τ action, then the corresponding session contract can perform it as well. If it is a synchronization, by definition of the semantics it involves two threads with complementary prefixes $\alpha_1 @ \ldots @ \alpha_n @ \rho'$ and $\overline{\alpha_1} @ \ldots \overline{\alpha_n} @ \sigma'$. By inductive hypothesis $\rho \parallel \sigma \xrightarrow{*}_{\mathsf{SC}} \rho' \parallel \sigma'$, hence the thesis follows since ρ' and σ' can match the synchronization. □

Appendix A.2. Proofs of Section 4

Proof of Proposition 11
Like $\dashv\!\Vert^{\mathsf{R}}$, $\dashv\!\Vert$ can be coinductively defined, since, by Corollary 4, $\dashv\!\Vert^{\mathsf{R}} = \dashv\!\Vert^{\mathsf{S}} = \dashv\!\Vert$. So $\dashv\!\Vert = \bigcap_n \dashv\!\Vert_n$ where $\dashv\!\Vert_n = \dashv\!\Vert^{\mathsf{R}}_n$.
For the definition of $\dashv\!\Vert_n$ see the discussion after Lemma 27.

We shall prove that

$$\forall n.[(\rho \dashv\!\Vert_n \sigma \text{ and } \overline{\sigma} \dashv\!\Vert_n \sigma') \text{ implies } \rho \dashv\!\Vert_n \sigma'] \tag{A.1}$$

So, from $\rho \dashv\!\Vert \sigma$ and $\overline{\sigma} \dashv\!\Vert \sigma'$ we have that $\forall n.\rho \dashv\!\Vert_n \sigma$ and $\forall n.\overline{\sigma} \dashv\!\Vert_n \sigma'$ and hence, by A.1 we can get $\forall n.\rho \dashv\!\Vert_n \sigma'$, that is $\rho \dashv\!\Vert \sigma'$.

We show now A.1 by induction on n.

The base case is trivial. Let then $n > 0$ with $\rho \dashv\!\Vert_n \sigma$ and $\overline{\sigma} \dashv\!\Vert_n \sigma'$. We proceed by cases according to the possible shapes of ρ and σ in the definition of $\dashv\!\Vert_n$.

$\rho = \mathbf{1}$ Immediate.

$\rho = \sum_{i\in I} \alpha_i.\rho_i$, $\sigma = \sum_{j\in J} \overline{\alpha}_j.\sigma_j$ **and** $\exists k \in I \cap J.\ \rho_k \dashv\!\shortmid_{n-1} \sigma_k$
We have then that $\overline{\sigma} = \bigoplus_{j\in J} \alpha_j.\overline{\sigma}_j$. So, by $\overline{\sigma} \dashv\!\shortmid_n \sigma'$ and by definition of $\dashv\!\shortmid_n$ we have that $\sigma' = \sum_{h\in H} \overline{\alpha}_h.\sigma'_h$, $J \subseteq H$ and $\forall j \in J.\ \overline{\sigma}_j \dashv\!\shortmid_{n-1} \sigma'_j$. By the induction hypothesis we can hence get that $\exists k \in (I \cap J) \subseteq (I \cap H)$ such that $\rho_k \dashv\!\shortmid_{n-1} \sigma'_k$, that means, by definition, that $\rho \dashv\!\shortmid_n \sigma'$.

$\rho = \bigoplus_{i\in I} \overline{\alpha}_i.\rho_i$, $\sigma = \sum_{j\in J} \alpha_j.\sigma_j$, $I \subseteq J$ **and** $\forall k \in I.\ \rho_k \dashv\!\shortmid_{n-1} \sigma_k$
We have then that $\overline{\sigma} = \bigoplus_{j\in J} \overline{\alpha}_j.\overline{\sigma}_j$. So, by $\overline{\sigma} \dashv\!\shortmid_n \sigma'$ and by definition of $\dashv\!\shortmid_n$ we have that $\sigma' = \sum_{h\in H} \alpha_h.\sigma'_h$, $J \subseteq H$ and $\forall j \in J.\ \overline{\sigma}_j \dashv\!\shortmid_{n-1} \sigma'_j$. By the induction hypothesis we can hence get that $\forall k \in I \subseteq J \subseteq H.\ \rho_k \dashv\!\shortmid_{n-1} \sigma'_k$, that means, by definition, that $\rho \dashv\!\shortmid_n \sigma'$.

$\rho = \sum_{i\in I} \overline{\alpha}_i.\rho_i$, $\sigma = \bigoplus_{j\in J} \alpha_j.\sigma_j$, $I \supseteq J$ **and** $\forall k \in J.\ \rho_k \dashv\!\shortmid_{n-1} \sigma_k$
We have then that $\overline{\sigma} = \sum_{j\in J} \overline{\alpha}_j.\overline{\sigma}_j$. So, by $\overline{\sigma} \dashv\!\shortmid_n \sigma'$ and by definition of $\dashv\!\shortmid_n$ we have to take into account two cases.

- $\sigma' = \sum_{h\in H} \alpha_h.\sigma'_h$ **and** $\exists k \in J \cap H.\ \sigma_k \dashv\!\shortmid_{n-1} \sigma'_k$
  By the induction hypothesis we can get that $\exists k \in (J \cap H) \subseteq (I \cap H)$ such that $\rho_k \dashv\!\shortmid_{n-1} \sigma'_k$, that means, by definition, that $\rho \dashv\!\shortmid_n \sigma'$.
- $\sigma' = \bigoplus_{h\in H} \alpha_h.\sigma'_h$, $J \supseteq H$ **and** $\forall h \in H.\ \sigma_h \dashv\!\shortmid_{n-1} \sigma'_h$
  By the induction hypothesis we can get that $\forall h \in H \subseteq J \subseteq I.\ \rho_h \dashv\!\shortmid_{n-1} \sigma'_h$, that means, by definition, that $\rho \dashv\!\shortmid_n \sigma'$. □

Proof of Theorem 12
(i) ($\Rightarrow$) By contraposition, assume that $\overline{\sigma} \not\dashv\!\shortmid \sigma'$. Since $\overline{\sigma} \dashv\!\shortmid \sigma$ by Proposition 10, then by definition of $\preccurlyeq_s$ we have $\sigma \not\preccurlyeq_s \sigma'$.
($\Leftarrow$) Let $\overline{\sigma} \dashv\!\shortmid \sigma'$. If $\rho \dashv\!\shortmid \sigma$, then from $\overline{\sigma} \dashv\!\shortmid \sigma'$, we get $\rho \dashv\!\shortmid \sigma'$ by Proposition 11, and therefore $\sigma \preccurlyeq_s \sigma'$ by definition.
(ii) ($\Rightarrow$) By contraposition, assume that $\sigma' \not\dashv\!\shortmid \overline{\sigma}$. Since $\sigma \dashv\!\shortmid \overline{\sigma}$ by Proposition 10, then by definition of $\preccurlyeq_c$ we have $\sigma \not\preccurlyeq_c \sigma'$.
($\Leftarrow$) Let $\sigma' \dashv\!\shortmid \overline{\sigma}$. If $\sigma \dashv\!\shortmid \rho$, then from $\sigma' \dashv\!\shortmid \overline{\sigma}$, we get $\sigma' \dashv\!\shortmid \rho$ by Proposition 11, and therefore $\sigma \preccurlyeq_c \sigma'$ by definition.
(iii) From Item (i) we have $\sigma \preccurlyeq_s \sigma'$ iff $\overline{\sigma} \dashv\!\shortmid \sigma'$. From Item (ii) we have $\overline{\sigma'} \preccurlyeq_c \overline{\sigma}$ iff $\overline{\sigma} \dashv\!\shortmid \overline{\overline{\sigma'}}$. The thesis follows since $\overline{\overline{\sigma'}} = \sigma'$.
(iv) From Items (i) and (ii) and decidability of $\overline{\sigma} \dashv\!\shortmid \sigma'$. □

Proof sketch of Proposition 16
For the proof of the antisymmetric property we cannot rely directly on the definition of $\preccurlyeq_s$, since from $\sigma \preccurlyeq_s \sigma'$ and $\sigma' \preccurlyeq_s \sigma$ we can only infer that σ and σ' have the same set of clients. This does not trivially imply that $\sigma = \sigma'$.
We can proceed, instead, roughly as follows. Let $\sigma, \sigma' \in \mathbf{rs}\mathsf{C}$ be such that $\sigma \preccurlyeq_s \sigma'$ and $\sigma' \preccurlyeq_s \sigma$. By completeness of $\blacktriangleright$ we get $\blacktriangleright \sigma \ll \sigma'$ and $\blacktriangleright \sigma' \ll \sigma$. By having such derivations, we can infer that in each of them no rule ($\oplus \cdot + \text{-} \preccurlyeq_s$) can be present. Moreover, in each application of rule ($+ \cdot + \text{-} \preccurlyeq_s$) or rule ($\oplus \cdot \oplus \text{-} \preccurlyeq_s$), we have necessarily that $J = \emptyset$. Out of that we can infer $\sigma = \sigma'$. □

Appendix A.3. Proofs of Section 5

Proof sketch of Lemma 19

($\Rightarrow$) A derivation of $\rhd \ \rho \backsim \sigma$ containing no occurrence of rule (Hyp) can be turned in a derivation of $\rhd_\infty \ \rho \backsim \sigma$ by erasing all the environments.
Otherwise, we observe that if an occurrence $\Gamma', \rho' \backsim \sigma' \ \rhd \ \rho' \backsim \sigma'$ of rule (Hyp) occurs in a given derivation of $\rhd \ \rho \backsim \sigma$, then the expression $\rho' \backsim \sigma'$ is discharged by a rule (namely, it occurs in the environment of the premise(s) but not in the one of the conclusion; we dub the occurrence of the rule a *discharging occurrence*) present in the path from the root to (Hyp). Hence a derivation of $\rhd_\infty \ \rho \backsim \sigma$ can be built by performing infinitely many times the replacement of all the occurences of rule (Hyp) with the subderivations having their corresponding discharging occurrences as roots. The infinite tree obtained by applying such a procedure and then erasing all the environments is a derivation of $\rhd_\infty \ \rho \backsim \sigma$.
($\Leftarrow$) Take an infinite derivation of $\rhd_\infty \ \rho \backsim \sigma$ (for a finite one the result is immediate). Since all contracts in the premises of a rule are subexpressions of those in the conclusion, and contracts are regular expressions, then in an infinite branch there must be at least a judgment occurring infinitely many times. It is not difficult hence to get a derivation of $\rhd \ \rho \backsim \sigma$ by replacing all the expressions of the form $\rho' \backsim \sigma'$ occurring for the second time (starting from the root) in a branch, by rules (Hyp), and by inserting environments accordingly. □

Proof of Lemma 20

First observe that:

(1) The recursive calls in lines -1-, -2-, 19- do leave unaltered the arguments A and F; so do those in lines -7-, -11-, -15-, -20- when the results A_0 and F_0 of the previous call are equal, respectively, to its first and second argument. In the depth-first search for a proof in $\rhd_\infty$ all these calls do correspond to trying and finding a subproof for a further pair of subterms of the initial ρ and σ once the search for a proof for the current pair has been (successfully or unsuccessfully) completed. This means that the overall number of such calls during the execution of the algorithm cannot exceed the number of pairs of subterms of the initial ρ and σ, so it is $\mathcal{O}(n^2)$;

(2) Also the recursive calls in lines -16-, -17- do leave unaltered the arguments A and F, but there cannot be more than two consecutive calls of this sort. This implies that the overall number of such calls cannot exceed two times the number of all the other calls;

(3) in all the other recursive calls the cardinality of $\mathsf{A} \cup \mathsf{F}$ strictly increases.

Hence, by the above and by the termination conditions of the algorithm, the overall number of calls cannot exceed $(n^2 + n^2) + (2 \times (n^2 + n^2))$.

We now observe that both the element-insertion and the membership-check operations for the sets A and F can be performed in $\mathcal{O}(1)$. It is in fact enough to build, before running the algorithm, two boolean vectors indexed by all the pairs of subterms of the initial ρ and σ. Building such vectors takes $\mathcal{O}(n^2)$ time.

Before concluding, we also observe - for what concerns the membership checks - that without loss of generality we can assume the initial ρ and σ to be lenght-minimal (where a contract γ is *lenght-minimal* whenever, for any contract δ such that γ and δ represent the same regular tree, the lenght of γ is not greater than that of δ). It can hence be checked that lenght-minimality is mantained by the argument of any recursive call but those in lines -16- and -17-, which are however performed after the membership check.

The complexity of the algorithm is hence $\mathcal{O}(n^2)$. □

Appendix A.4. Proofs of Section 6

Proof sketch of Theorem 23

The definition of the relation R is quite convoluted, hence we will not spell out it, but we present the main ideas below. Essentially, one starts by adding to R all pairs of the form $(\langle\,\rangle \ltimes \rho \parallel \langle\,\rangle \ltimes \sigma, [\![\rho \parallel \sigma]\!])$. This takes into account client/server pairs that have not performed any action. It is easy to check by induction on the definition of the translation function that the condition on barbs is satisfied.

In order to satisfy the other conditions, we need to add pairs obtained by performing actions. On the side of transformed contracts, performing actions just means underlining prefixes (there is no need to explicitly unfold recursion since we are using an equi-recursive interpretation of recursion), with the constraints that a prefix of an underlined prefix is underlined, and that in choices at most one alternative can be underlined.

A given transformed contract $X \parallel X'$ with underlined prefixes is in the relation R with multiple retractable client/server pairs, each of the form $\mathrm{H} \ltimes \rho \parallel \mathrm{H}' \ltimes \rho'$. The possible contracts with history $\mathrm{H} \ltimes \rho$ (resp. $\mathrm{H}' \ltimes \rho'$) are derived from X (resp. X') by applying the following procedure recursively to the top element of X, until all the underlined prefixes have been removed.

- if $X = \underline{\alpha}.X_1$ then a $\circ$ is pushed on the history, and the procedure applied to X_1;
- if $X = \underline{\tau}.X_1 + \sigma$ then the procedure is applied to X_1;
- if $X = \underline{\alpha}.X_1 + \sigma'$ then the procedure is applied to X_1. Note that here $\sigma' = \sum_{i \in I} \pi_i.\sigma_i$, where each π_i is an action α. In fact, the conditions on underlines described above forbid to have underlined actions, while the form of terms produced by the translation function forbids to have τs. Nevertheless, there are multiple possibilities for the term to push in the history, one for each subset of the alternatives in I. Formally, we have a possibility for each term of the form $\sum_{i \in J} \pi_i.\sigma_i$ with $J \subseteq I$, where the choice on no elements denotes $\circ$. Branches which are removed (that is, in $I \setminus J$) correspond to alternatives that have already been tried and discarded. The actual term to push is the only contract $\hat{\sigma}$ such that $[\![\hat{\sigma}]\!] = \sum_{i \in J} \pi_i.\sigma_i$. Essentially, it is obtained by translating back τ-prefixed choices into internal choices.

It is easy to check that the resulting relation R is a simulation. □