Retractable and Speculative Contracts

Franco Barbanera, Ivan Lanese, Ugo De 'Liguoro

Retractable and Speculative Contracts ⋆

Franco Barbanera[1], Ivan Lanese[2], and Ugo de'Liguoro[3]

[1] Dipartimento di Matematica e Informatica, University of Catania
barba@dmi.unict.it
[2] Dipartimento di Informatica - Scienza e Ingegneria, University of Bologna/INRIA
ivan.lanese@gmail.com
[3] Dipartimento di Informatica, University of Torino
ugo.deliguoro@unito.it

Abstract. *Behavioral contracts* are abstract descriptions of the communications that clients and servers perform. 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 contracts, dealing respectively with *backtracking* and 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 polynomial time.

1 Introduction

Binary behavioral contracts [14,27,15] and binary session types [22] are abstractions of programs used to statically ensure that a client and a server interact successfully (see the survey in [24]). Along the years, the basic theory has been extended to deal with many features of clients and servers, such as exceptions [12], time [9], and so on. We consider here two new features: *backtracking*, allowing one to go back to previous stages of the interaction, and *speculative execution* [30], 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-recovery [1]. Speculative execution is used for efficiency reasons in different areas, from simulation [13], to thread-level optimization [31], to web services [16].

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

⋆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 FIR 1B8C1 of the University of Catania and Project FORMS 2015 of Turin. We thank Mariangiola Dezani-Ciancaglini for interesting discussions and useful suggestions.

semantics. Essentially, they add to the session contracts of [3,10] (called first-order session behaviors in [3]) an operator of *external choice among output* operations. 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 all the possibilities are explored concurrently, and it is enough for one of them to succeed to guarentee 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 [21].
- 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 polynomial 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.

A few preliminary results on the topic of this paper have been presented in a workshop paper [7], which considers *retractable session contracts*, i.e., retractable contracts without internal choice among inputs. The main result of [7] is the decidability of the compliance relation (while we study here also the complexity), which was obtained via an algorithm that we now know to be exponential. Here we present a more refined, polynomial one (Figure 7). In [7] the subcontract relation and the dual contract were not studied, and indeed the dual contract did not exist due to the absence of internal choice among inputs.

Proofs, additional examples and additional background material are available in a companion technical report [8].

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 session contracts of [3,10] (called first-order session behaviors in [3]), that we dub here SC, 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 [7], that we dub here $\mathbf{rC}$, simply adding internal choice among inputs. Basics of session contracts and retractable session contracts are recalled in the companion technical report [8].

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,*

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

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 $\sigma[\mathsf{rec}\, x.\sigma/x]$. 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 is not a source of confusion 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 $\mathbf{rs}\mathsf{C}$. 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, 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, which memorizes, for past choices, the alternatives that have been discharged and that can be tried upon rollback. We use '$\circ$' to stand for no-remaining-alternatives.

Definition 2 (Contracts with History). *Let* Histories *be the expressions generated by the grammar* $\mathrm{H} ::= \langle\,\rangle \mid \mathrm{H}{:}\sigma$*, where* $\sigma \in \mathbf{rsC} \cup \{\circ\}$ *and* $\circ \notin \mathbf{rsC}$*.* 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 $(\cdots(\langle\,\rangle{:}\sigma_1){:}\cdots){:}\sigma_k$.

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 use it to define a reduction semantics for pairs of contracts representing one client and one server (Definition 4).

Definition 3 (Semantics of Contracts with History).

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

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

In the transition rule for external choice $(+)$, the action α is executed, and the discharged branches in σ' are memorized. In internal choice $(\oplus)$, instead, the selection of one branch is represented by a label τ, and the history H is unchanged. When a single action 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:*

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

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

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

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

Remark 1. 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., [17,29,25,26]. In particular, the semantics would become a classic uncontrolled semantics (according to the terminology in [26]) 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 [26], specifying which rollback steps are allowed, and when. We discuss in Remark 2 the impact that removing the above control mechanisms would have on retractable contracts and on their theory.

Example 1. 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 day time, 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. 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 1, where we assume that

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

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.

$$
\begin{array}{rrcl}
 & \langle\,\rangle \ltimes \begin{array}{r}\overline{\texttt{QoSday}}.(\texttt{priceMed}.\overline{\texttt{ok}} \\ +\ \texttt{priceLow}.\overline{\texttt{ok}}) \\ +\ \overline{\texttt{QoSnight}}.\texttt{priceLow}.\overline{\texttt{ok}}\end{array} & \| & \langle\,\rangle \ltimes \sum_{\texttt{QoS}} \texttt{QoS}.\overline{\texttt{price}_{\texttt{QoS}}}.\texttt{ok} \\
\longrightarrow & \begin{array}{r}\langle\,\rangle : \overline{\texttt{QoSnight}}.\texttt{priceLow}.\overline{\texttt{ok}} \\ \ltimes \texttt{priceMed}.\overline{\texttt{ok}} + \texttt{priceLow}.\overline{\texttt{ok}}\end{array} & \| & \begin{array}{l}\langle\,\rangle : \sum_{\texttt{QoS}\neq\texttt{QoSday}} \texttt{QoS}.\overline{\texttt{price}_{\texttt{QoS}}}.\texttt{ok} \\ \ltimes\ \ \overline{\texttt{priceHigh}}.\texttt{ok}\end{array} \\
\longrightarrow & \langle\,\rangle \ltimes \overline{\texttt{QoSnight}}.\texttt{priceLow}.\overline{\texttt{ok}} & \| & \langle\,\rangle \ltimes \sum_{\texttt{QoS}\neq\texttt{QoSday}} \texttt{QoS}.\overline{\texttt{price}_{\texttt{QoS}}}.\texttt{ok} \\
\longrightarrow & \langle\,\rangle : \circ \ltimes \texttt{priceLow}.\overline{\texttt{ok}} & \| & \begin{array}{l}\langle\,\rangle : \sum_{\texttt{QoS}\neq\texttt{QoSday},\texttt{QoSnight}} \texttt{QoS}.\overline{\texttt{price}_{\texttt{QoS}}}.\texttt{ok} \\ \ltimes\ \ \overline{\texttt{priceLow}}.\texttt{ok}\end{array} \\
\longrightarrow & \langle\,\rangle : \circ : \circ \ltimes \overline{\texttt{ok}} & \| & \begin{array}{l}\langle\,\rangle : \sum_{\texttt{QoS}\neq\texttt{QoSday},\texttt{QoSnight}} \texttt{QoS}.\overline{\texttt{price}_{\texttt{QoS}}}.\texttt{ok} : \circ \\ \ltimes\ \ \texttt{ok}\end{array} \\
\longrightarrow & \langle\,\rangle : \circ : \circ : \circ \ltimes \mathbf{1} & \| & \begin{array}{l}\langle\,\rangle : \sum_{\texttt{QoS}\neq\texttt{QoSday},\texttt{QoSnight}} \texttt{QoS}.\overline{\texttt{price}_{\texttt{QoS}}}.\texttt{ok} : \circ : \circ \\ \ltimes\ \ \mathbf{1}\end{array}
\end{array}
$$

Fig. 1. An example of retractable interaction

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 a sequence (possibly empty) of actions uniquely identifying it.*

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

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

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).
In the LTS below, we use as labels actions $\alpha ::= a \mid \overline{a}$, sequences of actions $\beta ::= \alpha \mid \alpha\beta$, and complex labels $\beta_\tau ::= \tau \mid \beta \mid \beta, \mathbf{T}$.

$$
\begin{array}{lll}
\textit{(Fork)} & \textit{($\oplus$)} & \textit{(α)} \\
\alpha.\sigma + \sigma' \xrightarrow{\alpha,\sigma'} \alpha@\sigma & \alpha.\sigma \oplus \sigma' \xrightarrow{\tau} \alpha.\sigma & \alpha.\sigma \xrightarrow{\alpha} \alpha@\sigma
\end{array}
$$

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

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

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.

Because of rules (@-α), (@-α-T), and (@-τ), execution is allowed below an @ prefix. 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: $\overline{\alpha\beta} = \overline{\alpha}\overline{\beta}$.*

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

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

Example 2. A server provides access to multiple algorithms for SAT solving [35]. 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 [34], 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:

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

A sample computation proceeds as described in Figure 2, 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

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

Fig. 2. An example of speculative interaction

is managed. The first reduction in Figure 2 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 [3,10] 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 **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 $\dashv\!\vert^{\mathsf{R}}$).

i) The relation $\dashv\!\vert^{\mathsf{R}}$ on contracts with history is defined by:
$\mathrm{H}_1 \ltimes \rho \dashv\!\vert^{\mathsf{R}} \mathrm{H}_2 \ltimes \sigma$ *if, for each* $\mathrm{H}_1', \mathrm{H}_2', \rho', \sigma'$ *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 $\dashv\!\vert^{\mathsf{R}}$ on contracts is defined by: $\rho \dashv\!\vert^{\mathsf{R}} \sigma$ *if* $\langle\,\rangle \ltimes \rho \dashv\!\vert^{\mathsf{R}} \langle\,\rangle \ltimes \sigma$.

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

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

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

Fig. 3. System $\rhd$

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 $\dashv\!\!\|^{\mathsf{S}}$).
The relation $\dashv\!\!\|^{\mathsf{S}}$ on contracts is defined by:

$\rho \dashv\!\!\|^{\mathsf{S}} \sigma$ *if for each* $\mathbf{C}_\rho, \mathbf{C}_\sigma$ *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* $\mathbf{C}_\rho = \mathbf{C} \mid \alpha_1@...@\alpha_n@\mathbf{1}$

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 $\Gamma \rhd \rho \rightharpoonup \sigma$, where the environment Γ is a finite set of expressions of the form $\delta \rightharpoonup \gamma$, with $\rho, \sigma, \delta, \gamma \in \mathbf{rs}\mathsf{C}$. Axioms and rules are as in Figure 3.

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 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 $\rhd \rho \rightharpoonup \sigma$ instead of $\Gamma \rhd \rho \rightharpoonup \sigma$ when Γ is empty.

Derivability in system $\rhd$ is decidable, since it is syntax-directed and proof reconstruction does terminate.

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

We can prove the soundness and the completeness of the formal system $\rhd$ w.r.t. both the retractable and the speculative semantics (see [8] for the proofs).

Theorem 2 (Retractable Soundness and Completeness).

$$\rhd \rho \backsim \sigma \quad \textit{iff} \quad \rho \dashv\!\shortparallel^{\mathsf{R}} \sigma$$

Theorem 3 (Speculative Soundness and Completeness).

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

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 1 (Retractable and Speculative Compliances Coincide).

$$\dashv\!\shortparallel^{\mathsf{R}} \;=\; \dashv\!\shortparallel^{\mathsf{S}}$$

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

Corollary 2 (Compliance Decidability). *The relation $\dashv\!\shortparallel$ is decidable.*

Remark 2. We now discuss the impact on the compliance relation of the four mechanisms for controlling reversibility in the semantics of retractable client/server pairs (see Remark 1). In particular, we analyze what would happen by dropping each one of them in isolation:

Drop "Not all reductions 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 Subsect. 3.1) of session contracts any more. The case we consider is strictly more general, since we allow for both retractable and unretractable choices.

Drop 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.

Drop "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.

Drop "in choices the chosen path is not memorized": 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 the case we consider is strictly more general.

3.1 Conservativity Results

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 [3,10], and on the retractable session contracts (**r**C) as defined in [7].

As previously said, it is not difficult to check that session contracts SC are a subset of retractable session contracts **r**C, which, in turn, are a subset of the contracts **rs**C we are presently investigating, namely: $\mathsf{SC} \subsetneq$ **r**$\mathsf{C} \subsetneq$ **rs**C. Obviously the strict inclusion $\mathsf{SC} \subsetneq$ **rs**C is not enough, by itself, to guarantee the retractable and speculative operational semantics for **rs**C to be conservative extensions of the operational semantics of SC. We prove that it is so in the following Proposition 1. 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 in SC (recalled in the companion technical report [8]).

Proposition 1 (Operational Semantics Conservativity). *Let* $\rho, \sigma \in \mathsf{SC}$.

i) $\rho \parallel \sigma \xrightarrow{*}_{\mathsf{SC}} \rho' \parallel \sigma'$ *iff* $\mathrm{H}_1 \ltimes \rho \parallel \mathrm{H}_2 \ltimes \sigma \xrightarrow{*}_f \mathrm{H}'_1 \ltimes \rho' \parallel \mathrm{H}'_2 \ltimes \sigma'$ *for some* $\mathrm{H}_1, \mathrm{H}_2, \mathrm{H}'_1$ *and* H'_2

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

where $\longrightarrow_{\mathsf{SC}}$ *denotes the reduction relation on* SC *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, the compliance relation for session contracts SC is the restriction of the compliance relation $\dashv\!\!\parallel$ for our contracts to pairs of session contracts SC, and similarly for the restriction of $\dashv\!\!\parallel$ to retractable session contracts **r**C.

Proposition 2 (Compliances Conservativity).

i) Let $\rho, \sigma \in \mathsf{SC}$*:* $\rho \dashv\!\!\parallel_{\mathsf{SC}} \sigma$ *iff* $\rho \dashv\!\!\parallel \sigma$

ii) Let $\rho, \sigma \in$ **r**C*:* $\rho \dashv\!\!\parallel_{\mathbf{rC}} \sigma$ *iff* $\rho \dashv\!\!\parallel \sigma$

4 Duality and the Subcontract Relation

Unlike the retractable session contracts of [7], in the present setting it is possible to get a natural notion of *duality*. The dual $\overline{\sigma}$ of an element σ of $\mathbf{rs}\mathsf{C}$ is obtained, as for session contracts, by interchanging any name a with $\overline{a}$ and $+$ with $\oplus$.

The notion of dual contract allows one to combine pairs of contracts in the compliance relation, as follows:

Proposition 3. *For any $\rho, \sigma, \sigma' \in \mathbf{rs}\mathsf{C}$, $\rho \dashv\!\shortmid \sigma$ and $\overline{\sigma} \dashv\!\shortmid \sigma'$ imply $\rho \dashv\!\shortmid \sigma'$*

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*. Such a relation may be used for implementing contract-based query engines (see [28] for a detailed discussion). An analogous subcontract relation, denoted $\preccurlyeq_c$, can be defined for clients.

Definition 11 (Subcontract Relations for Servers and for Clients). *Let $\sigma, \sigma' \in \mathbf{rs}\mathsf{C}$. We define*

i) $\sigma \preccurlyeq_s \sigma' \quad \triangleq \quad \forall \rho \in \mathbf{rs}\mathsf{C}\ [\, \rho \dashv\!\shortmid \sigma \text{ implies } \rho \dashv\!\shortmid \sigma' \,]$
ii) $\sigma \preccurlyeq_c \sigma' \quad \triangleq \quad \forall \rho \in \mathbf{rs}\mathsf{C}\ [\, \sigma \dashv\!\shortmid \rho \text{ implies } \sigma' \dashv\!\shortmid \rho \,]$

Using Proposition 3 we can characterize both $\preccurlyeq_s$ and $\preccurlyeq_c$ in terms of duality and compliance, relate them and get their decidability.

Theorem 4. *For any $\sigma, \sigma' \in \mathbf{rs}\mathsf{C}$:*

i) $\sigma \preccurlyeq_s \sigma'$ *iff* $\overline{\sigma} \dashv\!\shortmid \sigma'$
ii) $\sigma \preccurlyeq_c \sigma'$ *iff* $\sigma' \dashv\!\shortmid \overline{\sigma}$
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 simply 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, that is it is a possible server, and it is smaller than all the other possible servers.

Proposition 4 (Dual as a Least Element w.r.t. $\preccurlyeq_s$). *Let $\rho \in \mathbf{rs}\mathsf{C}$. Then $\overline{\rho}$ is a server for ρ, namely $\rho \dashv\!\shortmid \overline{\rho}$, and more precisely it is a least element in the set of the servers of ρ, that is,*

$$\forall \sigma \in \mathbf{rs}\mathsf{C}\text{:}\ \ \rho \dashv\!\shortmid \sigma \ \ \text{implies}\ \ \overline{\rho} \preccurlyeq_s \sigma$$

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 Prop. 5.

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 $\preccurlyeq_s$.

$$\frac{}{\Gamma \blacktriangleright \mathbf{1} \ll \sigma'}\ (\text{Ax-}\preccurlyeq_s) \qquad \frac{}{\Gamma, \sigma \ll \sigma' \blacktriangleright \sigma \ll \sigma'}\ (\text{Hyp-}\preccurlyeq_s) \qquad \frac{\Gamma, \alpha.\sigma_1 \oplus \sigma_2 \ll \alpha.\sigma_1' + \sigma_2' \blacktriangleright \sigma_1 \ll \sigma_1'}{\Gamma \blacktriangleright \alpha.\sigma_1 \oplus \sigma_2 \ll \alpha.\sigma_1' + \sigma_2'}\ (\oplus\cdot+\text{-}\preccurlyeq_s)$$

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

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

Fig. 4. The formal system $\blacktriangleright$

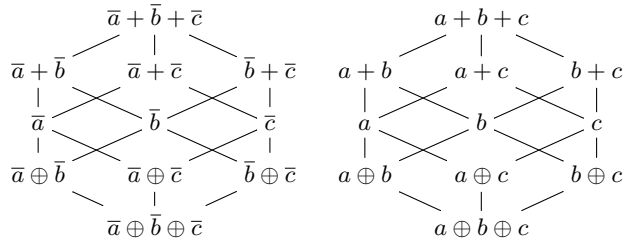

Fig. 5. Subcontract preorder: a sample

Definition 12 (Formal System for Subcontract $\blacktriangleright$). *Judgments in the formal system $\blacktriangleright$ 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 as in Figure 4.*

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

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

Theorem 5 (Soundness and Completeness of $\blacktriangleright$). $\blacktriangleright \sigma \ll \sigma'$ *iff* $\sigma \preccurlyeq_s \sigma'$

System $\blacktriangleright$ can be used to show that $\preccurlyeq_s$ is a partial order and hence, by antisymmetry, $\overline{\rho}$ is also the minimum server of ρ: it is minimal, hence there is no smaller server, and there is a unique minimal.

Proposition 5. *$\preccurlyeq_s$ is a partial order $\wedge$ $\forall \rho \in \mathbf{rsC}$, $\overline{\rho}$ is the minimum server of ρ.*

The structure of the partial order is shown in Figure 5, where the relations between terms with a unique choice among actions a, b, c, $\overline{a}$, $\overline{b}$ and $\overline{c}$ are pictured.

Remark 3. Analogously to what done in Subsect. 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).

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

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

Fig. 6. The non-well founded system $\rhd_\infty$

5 Complexity Issues

One can define a decision procedure for compliance as the recursive proof-search algorithm obtained by reading *bottom-up* the rules of the formal system for compliance in Figure 3. A similar algorithm is described in [7]. We show below that such an algorithm is strictly exponential.

To show it, roughly, it is possible to adapt the example presented in [20](§11) concerning 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.

$$\rho_0 = a + b \qquad \rho_{n+1} = \mathsf{rec}\, x.a.x + b.\rho_n$$
$$\sigma_0 = \mathsf{rec}\, x.\overline{a}.x \qquad \sigma_{n+1} = \overline{a}.\sigma_n \oplus \mathsf{rec}\, x.\overline{b}.x$$

As for the example in [20], 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, for any n, $\rho_n \dashv\Vdash \sigma_n$. By recursive breadth-first search, a derivation for $\rhd \rho_n \rightharpoonup \sigma_n$ is built in an actual exponential number of calls. Given n, the first part of the recursive-call tree looks as follows (where we denote by "**Ps**" the **P**roof-**s**earch algorithm)

$$\mathbf{Ps}(\emptyset \rhd \rho_n \rightharpoonup \sigma_n)$$
$$\mathbf{Ps}(\Gamma_1 \rhd \rho_n \rightharpoonup \sigma_{n-1}) \qquad \mathbf{Ps}(\Gamma_2 \rhd \rho_{n-1} \rightharpoonup \sigma_n)$$
$$\mathbf{Ps}(\Gamma_3 \rhd \rho_n \rightharpoonup \sigma_{n-2})\quad \underline{\mathbf{Ps}(\Gamma_4 \rhd \rho_{n-1} \rightharpoonup \sigma_{n-1})} \qquad \underline{\mathbf{Ps}(\Gamma_5 \rhd \rho_{n-1} \rightharpoonup \sigma_{n-1})}\quad \mathbf{Ps}(\Gamma_6 \rhd \rho_{n-2} \rightharpoonup \sigma_n)$$
$$\cdots\cdots \quad \text{etc.}$$

where $\Gamma_4 = \{\rho_n \rightharpoonup \sigma_n, \rho_n \rightharpoonup \sigma_{n-1}\} \neq \{\rho_n \rightharpoonup \sigma_n, \rho_{n-1} \rightharpoonup \sigma_n\} = \Gamma_5$. So, any call of the shape $\mathbf{Ps}(\Gamma \rhd \rho_k \rightharpoonup \sigma_k)$ produces two calls $\mathbf{Ps}(\Gamma' \rhd \rho_{k-1} \rightharpoonup \sigma_{k-1})$ and $\mathbf{Ps}(\Gamma'' \rhd \rho_{k-1} \rightharpoonup \sigma_{k-1})$ with $\Gamma' \neq \Gamma''$; overall there are at least 2^n calls.

However, the complexity of the compliance decision procedure can be drastically reduced down to a polynomial complexity as detailed below.

A polynomial decision algorithm.
We first define a non-well founded, but equivalent version of system $\rhd$.

Definition 13 (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 6 having $\rho \rightharpoonup \sigma$ as conclusion, and such that each finite branch ends with an instance of axiom* (Ax_∞).

Lemma 1 (Systems $\rhd$ and $\rhd_\infty$ are equivalent). $\rhd \rho \rightharpoonup \sigma$ *iff* $\rhd_\infty \rho \rightharpoonup \sigma$

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

$\mathbf{P}(\mathsf{A},\mathsf{F},[\,],\mathsf{b}) = (\mathsf{A},\mathsf{F},\mathsf{b})$
$\mathbf{P}(\mathsf{A},\mathsf{F},(\rho \curlywedge \sigma):\mathsf{xs},\mathsf{b}) =$
-1- **if** $\rho = \mathbf{1}$ **then** $\mathbf{P}(\mathsf{A},\mathsf{F},\mathsf{xs},\mathsf{b})$
-2- **else if** $\rho \curlywedge \sigma \in \mathsf{A}$ **then** $\mathbf{P}(\mathsf{A},\mathsf{F},\mathsf{xs},\mathsf{b})$
-3- **else if** $\rho \curlywedge \sigma \in \mathsf{F}$ **then** $(\mathsf{A},\mathsf{F},\mathbf{fail})$
-4- **else if** $\rho = \sum_{i\in I} \alpha_i.\rho_i$ **and** $\sigma = \sum_{j\in J} \overline{\alpha}_j.\sigma_j$ **and** $I \cap J = \{i_1,\ldots,i_n\}$
-5- **then let** $(\mathsf{A}_0,\mathsf{F}_0,\mathsf{b}_0) = \mathbf{P}^{+}(\mathsf{A}\cup\{\rho \curlywedge \sigma\},\mathsf{F},[\rho_{i_1} \curlywedge \sigma_{i_1} \ldots \rho_{i_n} \curlywedge \sigma_{i_n}],\mathsf{b})$
-6- **in if** b_0=**fail then** $(\mathsf{A}_0,\mathsf{F}_0,\mathbf{fail})$
-7- **else** $\mathbf{P}(\mathsf{A}_0,\mathsf{F}_0,\mathsf{xs},\mathsf{b}_0)$
-8- **else if** $\rho = \bigoplus_{i\in I} \overline{\alpha}_i.\rho_i$ **and** $\sigma = \sum_{j\in J} \alpha_j.\sigma_j$ **and** $I \subseteq J$ **and** $I = \{i_1,\ldots,i_n\}$
-9- **then let** $(\mathsf{A}_0,\mathsf{F}_0,\mathsf{b}_0) = \mathbf{P}(\mathsf{A}\cup\{\rho \curlywedge \sigma\},\mathsf{F},[\rho_{i_1} \curlywedge \sigma_{i_1} \ldots \rho_{i_n} \curlywedge \sigma_{i_n}],\mathsf{b})$
-10- **in if** b_0=**fail then** $(\mathsf{A}_0,\mathsf{F}_0,\mathbf{fail})$
-11- **else** $\mathbf{P}(\mathsf{A}_0,\mathsf{F}_0,\mathsf{xs},\mathsf{b}_0)$
-12- **else if** $\rho = \sum_{i\in I} \overline{\alpha}_i.\rho_i$ **and** $\sigma = \bigoplus_{j\in J} \alpha_j.\sigma_j$ **and** $I \supseteq J$ **and** $J = \{j_1,\ldots,j_n\}$
-13- **then let** $(\mathsf{A}_0,\mathsf{F}_0,\mathsf{b}_0) = \mathbf{P}(\mathsf{A}\cup\{\rho \curlywedge \sigma\},\mathsf{F},[\rho_{i_1} \curlywedge \sigma_{i_1} \ldots \rho_{j_n} \curlywedge \sigma_{j_n}],\mathsf{b})$
-14- **in if** b_0=**fail then** $(\mathsf{A}_0,\mathsf{F}_0,\mathbf{fail})$
-15- **else** $\mathbf{P}(\mathsf{A}_0,\mathsf{F}_0,\mathsf{xs},\mathsf{b}_0)$
-16- **else if** $\rho = \mathsf{rec}\,x.\rho'$ **then** $\mathbf{P}(\mathsf{A},\mathsf{F},(\{\mathsf{rec}\,x.\rho'/x\}\rho' \curlywedge \sigma):\mathsf{xs},\mathsf{b})$
-17- **else if** $\sigma = \mathsf{rec}\,x.\sigma'$ **then** $\mathbf{P}(\mathsf{A},\mathsf{F},(\rho \curlywedge \{\mathsf{rec}\,x.\sigma'/x\}\sigma'):\mathsf{xs},\mathsf{b})$
-18- **else** $(\mathsf{A},\mathsf{F}\cup\{\rho \curlywedge \sigma\},\mathbf{fail})$

and where

$\mathbf{P}^{+}(\mathsf{A},\mathsf{F},[\rho \curlywedge \sigma],\mathsf{b}) = \mathbf{P}(\mathsf{A},\mathsf{F},[\rho \curlywedge \sigma],\mathsf{b})$
$\mathbf{P}^{+}(\mathsf{A},\mathsf{F},(\rho \curlywedge \sigma):\mathsf{xs},\mathsf{b})=$
-19- **let** $(\mathsf{A}_0,\mathsf{F}_0,\mathsf{b}_0) = \mathbf{P}(\mathsf{A},\mathsf{F},[\rho \curlywedge \sigma],\mathsf{b})$ **in**
-20- **if** $\mathsf{b}_0 = \mathbf{fail}$ **then** $\mathbf{P}^{+}(\mathsf{A}\cup\mathsf{A}_0,\mathsf{F}\cup\mathsf{F}_0,\mathsf{xs},\mathbf{ok})$ **else** $(\mathsf{A}_0,\mathsf{F}_0,\mathsf{b}_0)$

Fig. 7. The polynomial decision procedure for compliance

In Figure 7 we present a decision algorithm $\mathbf{Decide}_{\dashv\!\shortparallel}$, based on the procedures $\mathbf{P}$ and $\mathbf{P}^{+}$. A run of the algorithm resembles a computation tree 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 **P**rovability procedure $\mathbf{P}$ is an adaptation of the concrete subtyping algorithm for recursive arrow and product types of [20](§10) to the present, more complex context. 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 [20](§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 $(+\cdot+)$ 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.

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

Proposition 6 (Complexity of Deciding Compliance/Subcontract). *Given two contracts $\rho, \sigma \in \mathbf{rsC}$, deciding whether $\rho \dashv\!\shortparallel \sigma$ (or $\rho \preccurlyeq_s \sigma$) holds has a complexity $\mathcal{O}(n^5)$, where n is the maximum size of ρ and σ.*

Remark 4. It is worth noticing that the polynomial decision procedure $\mathbf{Decide}_{\dashv\!\shortparallel}$ applies also to the formalism of retractable session contracts of [7] (in this case clauses at lines -8- and -12- are never used) and to the formalism of sessions contracts (some more clauses are never used).

6 Related Work and Conclusion

We have presented two conservative extensions of the session contracts of [2,3,10], a formalism interpreting session types [22] into a subset of contracts [14,27,15]. 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 w.r.t. the preliminary results about retractable session contracts in [7]. Another closely related work is [5,6], 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 (in [5,6] it is an internal decision of a participant), thus improving compliance; (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 [5,6].

Reversibility, generalizing backtracking by allowing one to go back to any past state, has also been studied in the setting of binary session types [32,33]. There however the emphasis is on defining the reversible engine, based on causal-consistent reversibility [26], 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 [18], 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 [4] a game-theoretical interpretation of the retractable session contracts of [7] has been provided. Such an interpretation is likely to extend to the retractable contracts presented here.

We plan also to investigate whether our approach can be extended to multiparty sessions [23]. An investigation of multi-party sessions with rollbacks and named checkpoints has been already undertaken in [19]. 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 [3,11]. It is hence worth studying the integration of backtracking (or speculative execution) and session delegation.

A last line of future work is the study of how to extract retractable or speculative contracts from actual software based on backtracking or on speculative parallelism, and how to propagate the results on contracts to the original software.

References

1. A. Avizienis, J.-C. Laprie, B. Randell, and C. E. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Trans. Dep. Sec. Comput.*, 1(1):11–33, 2004.
2. F. Barbanera and U. de'Liguoro. Two notions of sub-behaviour for session-based client/server systems. In *PPDP*, pages 155–164. ACM Press, 2010.
3. F. Barbanera and U. de'Liguoro. Sub-behaviour relations for session-based client/server systems. *MSCS*, 25(6):1339–1381, 2015.
4. F. Barbanera and U. de'Liguoro. A game interpretation of retractable contracts. In *COORDINATION*, volume 9686 of *LNCS*. Springer, 2016.
5. F. Barbanera, M. Dezani-Ciancaglini, and U. de'Liguoro. Compliance for reversible client/server interactions. In *BEAT*, volume 162 of *EPTCS*, pages 35–42, 2014.
6. F. Barbanera, M. Dezani-Ciancaglini, and U. de'Liguoro. Reversible client/server interactions. *Formal Asp. Comput.*, 28(4):697–722, 2016.
7. F. Barbanera, M. Dezani-Ciancaglini, I. Lanese, and U. de'Liguoro. Retractable contracts. In *PLACES 2015*, volume 203 of *EPTCS*, pages 61–72. Open Publishing Association, 2016.
8. F. Barbanera, I. Lanese, and U. de'Liguoro. Retractable and speculative contracts (TR). `http://www.cs.unibo.it/~lanese/tmp/TR-coord2017.pdf`, 2017.
9. M. Bartoletti et al. Compliance and subtyping in timed session types. In *FORTE*, volume 9039 of *LNCS*, pages 161–177. Springer, 2015.

10. G. T. Bernardi and M. Hennessy. Modelling session types using contracts. *Mathematical Structures in Computer Science*, 26(3):510–560, 2016.
11. G. T. Bernardi and M. Hennessy. Using higher-order contracts to model session types. *Logical Methods in Computer Science*, 12(2), 2016.
12. M. Carbone, K. Honda, and N. Yoshida. Structured interactional exceptions in session types. In *CONCUR*, volume 5201 of *LNCS*, pages 402–417. Springer, 2008.
13. C. D. Carothers, K. S. Perumalla, and R. Fujimoto. Efficient optimistic parallel simulations using reverse computation. *ACM Trans. Model. Comput. Simul.*, 9(3):224–253, 1999.
14. S. Carpineti, G. Castagna, C. Laneve, and L. Padovani. A formal account of contracts for Web Services. In *WS-FM*, number 4184 in LNCS, pages 148–162. Springer, 2006.
15. G. Castagna, N. Gesbert, and L. Padovani. A theory of contracts for web services. *ACM Trans. on Prog. Lang. and Sys.*, 31(5):19:1–19:61, 2009.
16. M. Dalla Preda et al. Graceful interruption of request-response service interactions. In *ICSOC*, volume 7084 of *LNCS*, pages 590–600. Springer, 2011.
17. V. Danos and J. Krivine. Reversible communicating systems. In *CONCUR*, volume 3170 of *LNCS*, pages 292–307. Springer, 2004.
18. E. de Vries, V. Koutavas, and M. Hennessy. Communicating transactions - (extended abstract). In *CONCUR*, volume 6269 of *LNCS*, pages 569–583. Springer, 2010.
19. M. Dezani-Ciancaglini and P. Giannini. Reversible multiparty sessions with checkpoints. In *EXPRESS/SOS'16*, volume 222 of *EPTCS*, pages 60–74, 2016.
20. V. Gapeyev, M. Y. Levin, and B. C. Pierce. Recursive subtyping revealed. *J. Funct. Program.*, 12(6):511–548, 2002.
21. C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.
22. K. Honda, V. T. Vasconcelos, and M. Kubo. Language primitives and type disciplines for structured communication-based programming. In *ESOP*, volume 1381 of *LNCS*, pages 22–138. Springer, 1998.
23. K. Honda, N. Yoshida, and M. Carbone. Multiparty asynchronous session types. In *POPL*, pages 273–284. ACM Press, 2008.
24. H. Hüttel et al. Foundations of session types and behavioural contracts. *ACM Comput. Surv.*, 49(1):3:1–3:36, 2016.
25. I. Lanese, C. A. Mezzina, and J.-B. Stefani. Reversibility in the higher-order π-calculus. *Theor. Comput. Sci.*, 625:25–84, 2016.
26. I. Lanese, C. A. Mezzina, and F. Tiezzi. Causal-consistent reversibility. *Bulletin of the EATCS*, 114, 2014.
27. C. Laneve and L. Padovani. The must preorder revisited: An algebraic theory for web services contracts. In *CONCUR*, volume 4703 of *LNCS*, pages 212–225. Springer, 2007.
28. L. Padovani. Contract-based discovery of web services modulo simple orchestrators. *Theoretical Computer Science*, 411:3328–3347, 2010.
29. I. C. C. Phillips and I. Ulidowski. Reversing algebraic process calculi. *J. of Logic and Alg. Progr.*, 73(1-2):70–96, 2007.
30. P. Prabhu, G. Ramalingam, and K. Vaswani. Safe programmable speculative parallelism. In *PLDI*, pages 50–61. ACM, 2010.
31. C. G. Quiñones et al. Mitosis compiler: An infrastructure for speculative threading based on pre-computation slices. In *PLDI*, pages 269–279. ACM, 2005.
32. F. Tiezzi and N. Yoshida. Towards reversible sessions. In *PLACES*, volume 155 of *EPTCS*, pages 17–24, 2014.
33. F. Tiezzi and N. Yoshida. Reversible session-based pi-calculus. *J. Log. Algebr. Meth. Program.*, 84(5):684–707, 2015.

34. L. Xu, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Satzilla: Portfolio-based algorithm selection for SAT. *J. Artif. Intell. Res. (JAIR)*, 32:565–606, 2008.
35. L. Zhang and S. Malik. The quest for efficient boolean satisfiability solvers. In *CAV*, volume 2404 of *LNCS*, pages 17–36. Springer, 2002.