Retractable Contracts*

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

Mariangiola Dezani-Ciancaglini
Dipartimento di Informatica, University of Torino, †
dezani@di.unito.it

Ivan Lanese
Dipartimento di Informatica - Scienza e Ingegneria, University of Bologna/INRIA, ‡
ivan.lanese@gmail.com

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

In calculi for modelling communication protocols, internal and external choices play dual roles. Two external choices can be viewed naturally as dual too, as they represent an agreement between the communicating parties. If the interaction fails, the past agreements are good candidates as points where to roll back, in order to take a different agreement. We propose a variant of contracts with synchronous rollbacks to agreement points in case of deadlock. The new calculus is equipped with a compliance relation which is shown to be decidable.

1 Introduction

In human as well as automatic negotiations, an interesting feature is the ability of rolling back to some previous point in case of failure, undoing previous choices and possibly trying a different path. *Roll-backs* are familiar to the users of web browsers, and so are also the troubles that these might cause during "undisciplined" interactions. Clicking the "back" button, or going to some previous point in the chronology when we are in the middle of a transaction, say the booking of a flight, can be as smart as dangerous. In any case, it is surely a behaviour that service programmers want to discipline. Also the converse has to be treated with care: a server discovering that an auxiliary service becomes available after having started a conversation could take advantage of it using some kind of rollback. However, such a server would be quite unfair if the rollback were completely hidden from the client.

Let us consider an example. A Buyer is looking for a bag ($\overline{\mathsf{bag}}$) or a belt ($\overline{\mathsf{belt}}$); she will decide how to pay, either by credit card ($\overline{\mathsf{card}}$) or by cash ($\overline{\mathsf{cash}}$), after knowing the price from a Seller. The Buyer behaviour can be described by the process:

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

*This work was partially supported by Italian MIUR PRIN Project CINA Prot. 2010LHT4KM and COST Action IC1201 BETTY.

†This author was partially supported by the Torino University/Compagnia San Paolo Project SALT.

‡This author was partially supported by the French ANR project REVER n. ANR 11 INSE 007 and COST Action IC1405.

§This author was partially supported by the Torino University/Compagnia San Paolo Project SALT.

S. J. Gay and J. Alglave (Eds.): Programming Language Approaches to
Concurrency- and Communication-Centric Software (PLACES 2015).
EPTCS 203, 2016, pp. **??**–72, doi:10.4204/EPTCS.203.5

where dot is sequential composition and $\oplus$ is internal choice. The Seller does not accept credit card payments for items of low price, like belts, but only for more expensive ones, like bags:

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

where $+$ is external choice. According to contract theory [6], Buyer is not compliant with Seller, since she can choose to pay the belt by card. Also, there is no obvious way to represent the buyer's will to be free in her decision about the payment and be compliant with a seller without asking the seller in advance. Nonetheless, when interacting with Seller, the buyer's decision is actually free at least in the case of purchase of a bag. For exploiting such a possibility the client (but also the server) should be able to tolerate a partial failure of her protocol, and to try a different path.

To this aim we add to (some) choices a possibility of rollback, in case the taken path fails to reach a success configuration. In this setting, choices among outputs are no more purely internal, since the environment may oblige to undo a wrong choice and choose a different alternative. For this reason, we denote choices between outputs which allow rollback as external, hence we use Buyer$'$ below instead of Buyer:

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

We thus explore a model of contract interaction in which synchronous rollback is triggered when client and server fail to reach an agreement.

In defining our model we build over some previous work reported in [2], where we have considered contracts with rollbacks. However, we depart from that model on three main aspects. First, in the present model rollback is used in a disciplined way to tolerate failures in the interaction, thus improving compatibility, while in [2] it is an internal decision of either client or server, which makes compatibility more difficult. Second, we embed checkpoints in the structure of contracts, avoiding explicit checkpoints. Third, we consider a stack of "pasts", called histories, instead of just one past for each participant, as in [2], thus allowing to undo many past choices looking for a successful alternative.

2 Contracts for retractable interactions

Our contracts can be obtained from the session behaviours of [1] or from the session contracts of [3] just adding external retractable choices between outputs.

Definition 2.1 (Retractable Contracts)**.** *Let* $\mathscr{N}$ (set of names) *be some countable set of symbols and* $\overline{\mathscr{N}} = \{\overline{a} \mid a \in \mathscr{N}\}$ (set of conames)*, with* $\mathscr{N} \cap \overline{\mathscr{N}} = \emptyset$*. The set* RC *of* **retractable contracts** *is defined as the set of the* **closed** *expressions generated by the following grammar,*

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

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

Note that recursion in RC 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 $\mathscr{N} \cup \overline{\mathscr{N}}$, with the convention $\overline{\alpha} = \begin{cases} \overline{a} & \text{if } \alpha = a, \\ a & \text{if } \alpha = \overline{a}. \end{cases}$

We write $\alpha_1.\sigma_1 + \alpha_2.\sigma_2$ for binary input/retractable output and $\overline{a}_1.\sigma_1 \oplus \overline{a}_2.\sigma_2$ for binary unretractable output. They are both commutative by definition. Also, $\overline{a}.\sigma$ may denote both unary retractable output and unary unretractable output. This is not a source of confusion since they have the same semantics.

From now on we call just *contracts* the expressions in RC. They are written by omitting all trailing **1**'s.

In order to deal with rollbacks we decorate contracts with histories, which memorise the alternatives in choices which have been discharged. We use '$\circ$' as a placeholder for *no-remaining-alternatives*.

Definition 2.2 (Contracts with histories)**.** *Let* Histories *be the expressions (referred to also as* stacks*) generated by the grammar:*

$$\boldsymbol{\gamma} ::= [\,] \mid \boldsymbol{\gamma} : \sigma$$

where $\sigma \in \mathsf{RC} \cup \{\circ\}$ *and* $\circ \notin \mathsf{RC}$*. Then the set of* contracts with histories *is defined by:*

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

We write just $\sigma_1 : \cdots : \sigma_k$ for the stack $(\cdots([\,] : \sigma_1) : \cdots) : \sigma_k$. With a little abuse of notation we use '$:$' also to concatenate histories, and to add contracts in front of histories.

We can now discuss the operational semantics of our calculus (Definition 2.3). The reduction rule for the internal choice ($\oplus$) is standard, but for the presence of the $\boldsymbol{\gamma} \prec \cdot$. Whereas, when reducing retractable choices (+), the discharged branches are memorised. When a single action is executed, the history is modified by adding a '$\circ$', intuitively meaning that the only possible branch has been tried and no alternative is left. Rule (rb) recovers the contract on the top of the stack, replacing the current one with it.

Definition 2.3 (LTS of Contracts with Histories)**.**

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

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

The interaction of a client with a server is modelled by the reduction of their parallel composition, that can be either forward, consisting of CCS style synchronisations 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 2.4 (LTS of Client/Server Pairs)**.** *We define the relation* $\longrightarrow$ *over pairs of contracts with histories by the following rules:*

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

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

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

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

We will use $\xrightarrow{*}$ and $\not\longrightarrow$ with the standard meanings.
Notice that, since '$\circ$' cannot synchronise with anything, in case a partner rolls back to a '$\circ$', it is forced to recover an *older* past (if any).

The following examples show the different behaviours of retractable and unretractable outputs. We decorate arrows with the name of the used reduction rule. As a first example we consider a possible reduction of the process discussed in the Introduction.

Example 2.5. As in the Introduction, let $\mathsf{Buyer}' = \overline{\mathsf{bag}}.\mathsf{price}.(\overline{\mathsf{card}} \oplus \overline{\mathsf{cash}}) + \overline{\mathsf{belt}}.\mathsf{price}.(\overline{\mathsf{card}} \oplus \overline{\mathsf{cash}})$ be a client and $\mathsf{Seller} = \mathsf{belt}.\overline{\mathsf{price}}.\mathsf{cash} + \mathsf{bag}.\overline{\mathsf{price}}.(\mathsf{card} + \mathsf{cash})$ a server; then

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

Example 2.6. Let $\rho = \mathsf{rec}\,x.(\overline{b}.x \oplus \overline{a}.c.x)$ and $\sigma = \mathsf{rec}\,x.(b.x + a.\overline{e}.x)$. The following reduction sequence leads the parallel composition of these contracts to a deadlock.

$$
\begin{array}{rrcl}
\rho \,\|\, \sigma & \xrightarrow{\tau} \quad [\,] \prec \overline{a}.c.\rho & \| & [\,] \prec \mathsf{rec}\,x.(b.x + a.\overline{e}.x) \\
 & \xrightarrow{\mathsf{comm}} \quad \circ \prec c.\rho & \| & b.\sigma \prec \overline{e}.\sigma \\
 & \xrightarrow{\mathsf{rbk}} \quad [\,] \prec \circ & \| & [\,] \prec b.\sigma \\
 & \not\longrightarrow & &
\end{array}
$$

Example 2.7. Let us now modify the above example by using retractable outputs in the client, so making the two contracts in parallel always reducible. The following reduction shows that there can be an infinite number of rollbacks in a sequence, even if it is not possible to have an infinite reduction containing only rollbacks. Notice how the stack keeps growing indefinitely.

Let $\rho = \mathsf{rec}\,x.(\overline{b}.x + \overline{a}.c.x)$ and $\sigma = \mathsf{rec}\,x.(b.x + a.\overline{e}.x)$.

$$\begin{array}{rcrcl}
\rho \parallel \sigma & \xrightarrow{\mathsf{comm}} & \overline{b}.\rho \prec c.\rho & \parallel & b.\sigma \prec \overline{e}.\sigma \\
& \xrightarrow{\mathsf{rbk}} & [\,] \prec \overline{b}.\rho & \parallel & [\,] \prec b.\sigma \\
& \xrightarrow{\mathsf{comm}} & \circ \prec \rho & \parallel & \circ \prec \sigma \\
& \xrightarrow{\mathsf{comm}} & \circ : \overline{b}.\rho \prec c.\rho & \parallel & \circ : b.\sigma \prec \overline{e}.\sigma \\
& \xrightarrow{\mathsf{rbk}} & \circ \prec \overline{b}.\rho & \parallel & \circ \prec b.\sigma \\
& \xrightarrow{\mathsf{comm}} & \circ : \circ \prec \rho & \parallel & \circ : \circ \prec \sigma \\
& & \vdots & &
\end{array}$$

3 Compliance

The compliance relation for standard contracts consists in requiring that, whenever no reduction is possible, all client requests and offers have been satisfied, i.e. the client is in the success state **1**. For retractable contracts we can adopt the same definition.

Definition 3.1 (Compliance Relation $\dashv\!\!\mid$)**.**

i) The relation $\dashv\!\!\mid$ on contracts with histories is defined by:

$$\boldsymbol{\delta} \prec \rho \dashv\!\!\mid \boldsymbol{\gamma} \prec \sigma \text{ whenever } \boldsymbol{\delta} \prec \rho \parallel \boldsymbol{\gamma} \prec \sigma \xrightarrow{*} \boldsymbol{\delta'} \prec \rho' \parallel \boldsymbol{\gamma'} \prec \sigma' \not\longrightarrow \text{ implies } \rho' = \mathbf{1} \text{ for any } \boldsymbol{\delta'}, \rho', \boldsymbol{\gamma'}, \sigma'.$$

ii) The relation $\dashv\!\!\mid$ on contracts is defined by:

$$\rho \dashv\!\!\mid \sigma \quad \text{if} \quad [\,] \prec \rho \dashv\!\!\mid [\,] \prec \sigma.$$

We now provide a formal system characterising compliance on retractable contracts.
The judgments are of the shape $\Gamma \rhd \rho \dashv \sigma$, where Γ is a set of expressions of the form $\rho' \dashv \sigma'$. We write $\rhd \rho \dashv \sigma$ when Γ is empty. The only non standard rule is rule $(+,+)$, which assures compliance of two retractable choices when they contain respectively a name and the corresponding coname followed by compliant contracts. This contrasts with the rules $(\oplus,+)$ and $(+,\oplus)$, where all conames in unretractable choices between outputs must have corresponding names in the choices between inputs, followed by compliant contracts.

Definition 3.2 (Formal System for Compliance)**.**

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

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

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

Notice that rule $(+,+)$ implicitly represents the fact that, in the decision procedure for two contracts made of retractable choices, the possible synchronising branches have to be tried, until either a successful one is found or all fail.

Example 3.3. Let us formally show that, for the Buyer' and Seller of the Introduction, we have $\mathsf{Buyer}' \dashv \mathsf{Seller}$.
For the sake of readability, let
$\Gamma' = \mathsf{Buyer}' \dashv \mathsf{Seller},\ \mathsf{price}.(\overline{\mathsf{card}} \oplus \overline{\mathsf{cash}}) \dashv \overline{\mathsf{price}}.(\mathsf{card} + \mathsf{cash})$ and $\Gamma'' = \Gamma',\ \overline{\mathsf{card}} \oplus \overline{\mathsf{cash}} \dashv \mathsf{card} + \mathsf{cash}$

$$\dfrac{\dfrac{\dfrac{}{\Gamma'' \rhd \mathbf{1} \dashv \mathbf{1}}\,(\textsc{Ax}) \qquad \dfrac{}{\Gamma'' \rhd \mathbf{1} \dashv \mathbf{1}}\,(\textsc{Ax})}{\Gamma' \rhd \overline{\mathsf{card}} \oplus \overline{\mathsf{cash}} \dashv \mathsf{card} + \mathsf{cash}}\,(\oplus,+)}{\mathsf{Buyer}' \dashv \mathsf{Seller} \rhd \mathsf{price}.(\overline{\mathsf{card}} \oplus \overline{\mathsf{cash}}) \dashv \overline{\mathsf{price}}.(\mathsf{card} + \mathsf{cash})}\,(+,+)}{\rhd \mathsf{Buyer}' \dashv \mathsf{Seller}}\,(+,+)$$

Example 3.4. The contracts of Example 2.7 can be formally proved to be compliant by means of the following derivation in our formal system. Actually such a derivation can be looked at as the result of the decision procedure implicitly described by the formal system.

$$\dfrac{\dfrac{}{\overline{b}.\rho + \overline{a}.c.\rho \dashv b.\sigma + a.\overline{e}.\sigma \rhd \rho \dashv \sigma}\,(\textsc{Hyp})}{\rhd \overline{b}.\rho + \overline{a}.c.\rho \dashv b.\sigma + a.\overline{e}.\sigma}\,(+,+)$$

In applying the rules we exploit the fact that we consider contracts modulo recursion fold/unfold.

We can show that derivability in this formal system is decidable, since it is syntax directed and it does not admit infinite derivations.

We denote by $\mathscr{D}$ a derivation in the system of Definition 3.2. The procedure **Prove** in Figure 1 clearly implements the formal system, that is it is straightforward to check the following

Fact 3.5.

i) **Prove**$(\Gamma \rhd \rho \dashv \sigma) \neq$ **fail** *iff* $\Gamma \rhd \rho \dashv \sigma$.

ii) **Prove**$(\Gamma \rhd \rho \dashv \sigma) = \mathscr{D} \neq$ **fail** *implies* $\overset{\mathscr{D}}{\Gamma \rhd \rho \dashv \sigma}$

Theorem 3.6. *Derivability in the formal system is decidable.*

Proof. By Fact 3.5, we only need to show that the procedure **Prove** always terminates. Notice that in all recursive calls **Prove**$(\Gamma, \rho \dashv \sigma \rhd \rho_k \dashv \sigma_k)$ inside **Prove**$(\Gamma \rhd \rho \dashv \sigma)$ the expressions ρ_k and σ_k are subexpressions of ρ and σ respectively (because of unfolding 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. □

In the remaining of this section we will show the soundness and the completeness of the formal system using some auxiliary lemmas.

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

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

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

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.** $\mathscr{D} = \mathbf{Prove}(\Gamma, \rho \dashv \sigma \rhd \rho_k \dashv \sigma_k) \neq \mathbf{fail}$
then $\dfrac{\mathscr{D}}{\Gamma \rhd \rho \dashv \sigma}\,(+,+)$ **else fail**

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

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

else fail

Figure 1: The procedure **Prove**.

Soundness It is useful to show that 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 3.7. *If* $\boldsymbol{\delta} \prec \rho' \parallel \boldsymbol{\gamma} \prec \sigma' \not\longrightarrow$, *then* $\boldsymbol{\delta} = \boldsymbol{\gamma} = [\,]$.

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

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

Then starting from two stacks containing the same number of elements, the reduction always produces two stacks containing the same number of elements. So $\boldsymbol{\delta} = [\,]$ implies $\boldsymbol{\gamma} = [\,]$ and vice versa. □

Next we state a lemma relating the logical rules of the formal system for compliance to the reduction rules; notice that the formers do not mention stacks in their judgments.

Lemma 3.8. *If the following is an instance of rule* $(+,+)$, *or* $(\oplus,+)$, *or* $(+,\oplus)$*:*

$$\frac{\Gamma, \rho \dashv \sigma \rhd \rho_1 \dashv \sigma_1 \quad \cdots \quad \Gamma, \rho \dashv \sigma \rhd \rho_n \dashv \sigma_n}{\Gamma \rhd \rho \dashv \sigma}$$

then for all $\boldsymbol{\delta}, \boldsymbol{\gamma}$ *and for all* $i = 1, \ldots, n$ *there exist* $\boldsymbol{\delta}_i, \boldsymbol{\gamma}_i$ *such that*

$$\boldsymbol{\delta} \prec \rho \parallel \boldsymbol{\gamma} \prec \sigma \;\xrightarrow{\;*\;}\; \boldsymbol{\delta}_i \prec \rho_i \parallel \boldsymbol{\gamma}_i \prec \sigma_i$$

and rule (rbk) *is not used, namely no rollback occurs.*

Proof. By inspection of the deduction and reduction rules. □

Theorem 3.9 (Soundness)**.** *If* $\rhd\ \rho \dashv \sigma$*, then* $\rho \dashv\!\shortmid \sigma$.

Proof. The proof is by contradiction. Assume $\rho \not\dashv\!\shortmid \sigma$. Then there is a reduction

$$[\,] \prec \rho \parallel [\,] \prec \sigma \xrightarrow{*} [\,] \prec \rho' \parallel [\,] \prec \sigma' \not\longrightarrow$$

with $\rho' \neq \mathbf{1}$. Note that both the histories are empty thanks to Lemma 3.7. We proceed by induction on the number n of steps in the reduction.

Let us consider the base case ($n = 0$). In this case $\rho \neq \mathbf{1}$ and there is no possible synchronization. Rule Ax is not applicable since $\rho \neq \mathbf{1}$. Rule Hyp is not applicable since Γ is empty. The other rules are not applicable otherwise we would have a possible synchronization.

Let us consider the inductive case. We have a case analysis on the topmost operators in ρ and σ. Let us start with the case where both topmost operators are retractable sums, i.e., $\rho = a.\rho_k + \rho''$ and $\sigma = \overline{a}.\sigma_k + \sigma''$. Thus,

$$[\,] \prec a.\rho_k + \rho'' \parallel [\,] \prec \overline{a}.\sigma_k + \sigma'' \xrightarrow{*} [\,] \prec \rho'' \parallel [\,] \prec \sigma'' \xrightarrow{*} [\,] \prec \rho' \parallel [\,] \prec \sigma' \not\longrightarrow .$$

By definition $\rho'' \not\dashv\!\shortmid \sigma''$, and since this requires a reduction of length $< n$, by inductive hypothesis $\not\rhd \rho'' \dashv \sigma''$. Also the above reduction begins by:

$$[\,] \prec a.\rho_k + \rho'' \parallel [\,] \prec \overline{a}.\sigma_k + \sigma'' \longrightarrow \rho'' \prec \rho_k \parallel \sigma'' \prec \sigma_k \xrightarrow{*} \rho'' \prec \rho_k' \parallel \sigma'' \prec \sigma_k' \longrightarrow [\,] \prec \rho'' \parallel [\,] \prec \sigma''$$

for some ρ_k', σ_k'. This implies, by the conditions on rule (rbk) and by Lemma 3.7:

$$[\,] \prec \rho_k \parallel [\,] \prec \sigma_k \xrightarrow{*} [\,] \prec \rho_k' \parallel [\,] \prec \sigma_k' \not\longrightarrow .$$

It follows that $\rho_k \not\dashv\!\shortmid \sigma_k$, and by inductive hypothesis $\not\rhd \rho_k \dashv \sigma_k$. We now claim that:

$$\not\rhd\ \rho'' \dashv \sigma'' \text{ and } \not\rhd\ \rho_k \dashv \sigma_k \text{ imply } \not\rhd\ a.\rho_k + \rho'' \dashv \overline{a}.\sigma_k + \sigma''$$

Toward a contradiction let us assume that $\rhd\ a.\rho_k + \rho'' \dashv \overline{a}.\sigma_k + \sigma''$; then the only applicable rule is $(+,+)$, which requires both

$$a.\rho_k + \rho'' \dashv \overline{a}.\sigma_k + \sigma'' \ \rhd\ \rho'' \dashv \sigma'' \text{ and } a.\rho_k + \rho'' \dashv \overline{a}.\sigma_k + \sigma'' \ \rhd\ \rho_k \dashv \sigma_k.$$

Because the only difference between these statements and $\rhd\ \rho'' \dashv \sigma''$ and $\rhd\ \rho_k \dashv \sigma_k$ is the assumption $a.\rho_k + \rho'' \dashv \overline{a}.\sigma_k + \sigma''$, which is used only in rule Hyp, there must be at least one branch of the derivation tree of $\rhd\ a.\rho_k + \rho'' \dashv \overline{a}.\sigma_k + \sigma''$ ending by such a rule. By Lemma 3.8 this implies that $[\,] \prec a.\rho_k + \rho'' \parallel [\,] \prec \overline{a}.\sigma_k + \sigma'' \xrightarrow{*} [\,] \prec a.\rho_k + \rho'' \parallel [\,] \prec \overline{a}.\sigma_k + \sigma''$ by a reduction never using rule (rbk). By definition this implies that $[\,] \prec a.\rho_k + \rho'' \dashv\!\shortmid [\,] \prec \overline{a}.\sigma_k + \sigma''$, contradicting the hypothesis.

All other cases are similar. □

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

Lemma 3.10. *If* $\boldsymbol{\delta} \prec \rho \dashv\!\shortmid \boldsymbol{\gamma} \prec \sigma$*, then* $\boldsymbol{\delta'} : \boldsymbol{\delta} \prec \rho \dashv\!\shortmid \boldsymbol{\gamma'} : \boldsymbol{\gamma} \prec \sigma$ *for all* $\boldsymbol{\delta'}$ *,* $\boldsymbol{\gamma'}$.

Proof. It suffices to show that

$$\boldsymbol{\delta} \prec \rho \dashv\!\!\!\shortmid \boldsymbol{\gamma} \prec \sigma \text{ implies } \rho' : \boldsymbol{\delta} \prec \rho \dashv\!\!\!\shortmid \boldsymbol{\gamma} \prec \sigma \text{ and } \boldsymbol{\delta} \prec \rho \dashv\!\!\!\shortmid \sigma' : \boldsymbol{\gamma} \prec \sigma$$

which we prove by contraposition.

Suppose that $\rho' : \boldsymbol{\delta} \prec \rho \not\dashv\!\!\!\shortmid \boldsymbol{\gamma} \prec \sigma$; then

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

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

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

Otherwise we have that

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

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

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

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

Similarly we can show that $\boldsymbol{\delta} \prec \rho \not\dashv\!\!\!\shortmid \sigma' : \boldsymbol{\gamma} \prec \sigma$ implies $\boldsymbol{\delta} \prec \rho \not\dashv\!\!\!\shortmid \boldsymbol{\gamma} \prec \sigma$. □

The following lemma gives all possible shapes of compliant contracts. It is the key lemma for the proof of completeness.

Lemma 3.11. *If $\rho \dashv\!\!\!\shortmid \sigma$, then 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* $\exists k \in I \cap J.\ \rho_k \dashv\!\!\!\shortmid \sigma_k$;
3. $\rho = \bigoplus_{i\in I} \overline{a}_i.\rho_i$, $\sigma = \sum_{j\in J} a_j.\sigma_j$, $I \subseteq J$ *and* $\forall k \in I.\ \rho_k \dashv\!\!\!\shortmid \sigma_k$;
4. $\rho = \sum_{i\in I} a_i.\rho_i$, $\sigma = \bigoplus_{j\in J} \overline{a}_j.\sigma_j$, $I \supseteq J$ *and* $\forall k \in J.\ \rho_k \dashv\!\!\!\shortmid \sigma_k$.

Proof. By contraposition and by cases of 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 $\rho_{k_i} \not\dashv\!\!\!\shortmid \sigma_{k_i}$ for $1 \leq i \leq n$. Then we get

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

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

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

by Lemma 3.10. Let $I' = I \setminus J$ and $J' = J \setminus I$. We can reduce $[\,] \prec \rho \parallel [\,] \prec \sigma$ only as follows:

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

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

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

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

Otherwise $I \subseteq J$ and $\rho_k \not\asymp \sigma_k$ for some $k \in I$. By reasoning as above we have

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

and

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

which imply

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

In both cases we conclude that $\rho \not\asymp \sigma$.

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

Theorem 3.12 (Completeness)**.** *If $\rho \dashv\!\asymp \sigma$, then $\rhd\ \rho \dashv \sigma$.*

Proof. By Theorem 3.6 each computation of **Prove**$(\rhd\ \rho \dashv \sigma)$ always terminates. By Lemma 3.11 and Fact 3.5, $\rho \dashv\!\asymp \sigma$ implies that **Prove**$(\rhd\ \rho \dashv \sigma) \neq$ **fail**, and hence $\rhd\ \rho \dashv \sigma$. □

4 Related work and conclusions

Since the pioneering work by Danos and Krivine [7], reversible concurrent computations have been widely studied. A main point is that understanding which actions can be reversed is not trivial in a concurrent setting, since there is no unique "last" action. Since [7], the most common notion of reversibility in concurrency is *causal-consistent* reversibility: any action can be undone if no other action depending on it has been executed (and not yet undone). The name highlights the relation with causality, which

makes the approach applicable even in settings where there is no unique notion of time, but makes it quite complex.

The first calculus for which a causal-consistent reversible extension has been defined is CCS in [7], using a stack of memories for each thread. Later, causal-consistent reversible extensions have been defined by Phillips and Ulidowski [13] for calculi definable by SOS rules in a general format (without mobility), using keys to bind synchronised actions together, and by Lanese et al. [10] for the higher-order π-calculus, using explicit memory processes to store history information and tags to track causality. A survey of causal-consistent reversibility can be found in [11].

In [9], Lanese et al. enrich the calculus of [10] with a fine-grained rollback primitive, showing the subtleties of defining a rollback operator in a concurrent setting. The first papers exploring reversibility in a context of sessions (see, e.g., [12] for a comparison between session types and contracts) are [14, 15], by Tiezzi and Yoshida. These papers define the semantics for reversible sessions by adapting the approach in [10], but do not consider compliance. Compliance has been first studied in [2]. We already discussed the differences between the present work and [2] in the Introduction.

A main point of our approach is that it exploits the fact that contracts describe sequential interactions (in a concurrent setting) to avoid the complexity of causal-consistent reversibility, allowing for a simpler semantics (compared, e.g., to the one of [9]).

Similarly to our approach, long running transactions with compensations, and in particular interacting transactions [16], allow to undo past agreements. In interacting transactions, however, a new possibility is tried when an exception is raised, not when an agreement cannot be found as in our case. Also, the possible options are sorted: first the normal execution, then the compensation. Finally, compliance of interacting transactions has never been studied. In the field of sessions, the most related works are probably the ones studying exceptions in binary sessions [5] and in multi-party sessions [4]. As for transactions, they aim at dealing with exceptions more than at avoiding to get stuck since an agreement cannot be found.

We plan to investigate whether our approach can be extended to multi-party sessions [8], the rationale being that parallelism is controlled by the global type, hence possibly part of the complexity due to concurrency can be avoided. The sub-behaviour relation induced by our notion of compliance is also worth being thoroughly studied.

Acknowledgments. We are grateful to the anonymous reviewers for their useful remarks.

References

[1] Franco Barbanera & Ugo de'Liguoro (2010): *Two Notions of Sub-behaviour for Session-based Client/Server Systems*. In: *PPDP*, ACM Press, pp. 155–164, doi:10.1145/1836089.1836109.

[2] Franco Barbanera, Mariangiola Dezani-Ciancaglini & Ugo de'Liguoro (2014): *Compliance for Reversible Client/Server Interactions*. In: *BEAT*, *EPTCS* 162, pp. 35–42, doi:10.4204/EPTCS.162.5.

[3] Giovanni Bernardi & Matthew Hennessy (2014): *Modelling Session Types using Contracts*. *Math. Struct. in Comp. Science*, doi:10.1017/S0960129514000243. To appear.

[4] Sara Capecchi, Elena Giachino & Nobuko Yoshida (2010): *Global Escape in Multiparty Sessions*. In: *FSTTCS*, *LIPIcs* 8, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, pp. 338–351, doi:10.4230/LIPIcs.FSTTCS.2010.338.

[5] Marco Carbone, Kohei Honda & Nobuko Yoshida (2008): *Structured Interactional Exceptions in Session Types*. In: *CONCUR*, *LNCS* 5201, Springer, pp. 402–417, doi:10.1007/978-3-540-85361-9_32.

[6] Giuseppe Castagna, Nils Gesbert & Luca Padovani (2009): *A Theory of Contracts for Web Services*. *ACM Trans. on Prog. Lang. and Sys.* 31(5), pp. 19:1–19:61, doi:10.1145/1538917.1538920.

[7] Vincent Danos & Jean Krivine (2004): *Reversible Communicating Systems*. In: *CONCUR*, *LNCS* 3170, Springer, pp. 292–307, doi:10.1007/978-3-540-28644-8_19.

[8] Kohei Honda, Nobuko Yoshida & Marco Carbone (2008): *Multiparty Asynchronous Session Types*. In: *POPL*, ACM Press, pp. 273–284, doi:10.1145/1328897.1328472.

[9] Ivan Lanese, Claudio Antares Mezzina, Alan Schmitt & Jean-Bernard Stefani (2011): *Controlling Reversibility in Higher-Order Pi*. In: *CONCUR*, *LNCS* 6901, Springer, pp. 297–311, doi:10.1007/978-3-642-23217-6_20.

[10] Ivan Lanese, Claudio Antares Mezzina & Jean-Bernard Stefani (2010): *Reversing Higher-Order Pi*. In: *CONCUR*, *LNCS* 6269, Springer, pp. 478–493, doi:10.1007/978-3-642-15375-4_33.

[11] Ivan Lanese, Claudio Antares Mezzina & Francesco Tiezzi (2014): *Causal-Consistent Reversibility*. *Bulletin of the EATCS* 114.

[12] Cosimo Laneve & Luca Padovani (2008): *The Pairing of Contracts and Session Types*. In: *Concurrency, Graphs and Models*, *LNCS* 5065, pp. 681–700, doi:10.1007/978-3-540-68679-8_42.

[13] Iain C. C. Phillips & Irek Ulidowski (2007): *Reversing Algebraic Process Calculi*. *J. of Logic and Alg. Progr.* 73(1-2), pp. 70–96, doi:10.1016/j.jlap.2006.11.002.

[14] Francesco Tiezzi & Nobuko Yoshida (2014): *Towards Reversible Sessions*. In: *PLACES*, *EPTCS* 155, pp. 17–24, doi:10.4204/EPTCS.155.3.

[15] Francesco Tiezzi & Nobuko Yoshida (2015): *Reversible session-based pi-calculus*. *J. Log. Algebr. Meth. Program.* 84(5), pp. 684–707, doi:10.1016/j.jlamp.2015.03.004.

[16] Edsko de Vries, Vasileios Koutavas & Matthew Hennessy (2010): *Communicating Transactions - (Extended Abstract)*. In: *CONCUR*, *LNCS* 6269, Springer, pp. 569–583, doi:10.1007/978-3-642-15375-4_39.