Modelling session types using contracts

GIOVANNI BERNARDI and MATTHEW HENNESSY

Mathematical Structures in Computer Science / Volume 26 / Special Issue 03 / March 2016, pp 510 - 560
DOI: 10.1017/S0960129514000243, Published online: 10 November 2014

Math. Struct. in Comp. Science (2016), *vol.* 26, *pp.* 510–560.
doi:10.1017/S0960129514000243 First published online 10 November 2014

Modelling session types using contracts

GIOVANNI BERNARDI and MATTHEW HENNESSY[†]

School of Statistics and Computer Science, The University of Dublin, Trinity College, Ireland
Email: bernargi@tcd.ie, matthew.hennessy@tcd.ie

Received 31 January 2013; revised 30 July 2013

Session types and contracts are two formalisms used to study client–server protocols. In this paper, we study the relationship between them. The main result is the existence of a fully abstract model of session types; this model is based on a natural interpretation of these types into a subset of contracts.

1. Introduction

Communication between processes in a distributed system often consists of a structured dialogue, following a protocol which specifies the format of the messages interchanged and, at least for binary communication, the direction of the messages. *Session types*, $\mathcal{ST}$, have been introduced as an approach to the static analysis of the participants of such dialogues. They allow structured sequences of non-uniform messages to be interchanged between the participants. For example, using the notation of Gay and Hole (2005), the type $![\texttt{Int}];?[\texttt{Real}];\textsc{end}$ specifies the output of a value of type Int followed by the input of a value of type Real, after which the dialogue is terminated. Flexibility in the permitted sequencing of messages by a process is accommodated by two choice operators; the *branching type* $\&\langle T_1, T_2 \rangle$ offers a choice to the partner in the dialogue between following either the protocol specified by the type T_1 or that specified by T_2. On the other hand, the *choice type* $\oplus\langle T_1, T_2 \rangle$ allows the process itself to follow either of the protocols specified by T_1 or T_2.

Sub-typing (Gay and Hole 2005), also increases the flexibility of the type system; intuitively $T_1 \preccurlyeq_{\text{ST}} T_2$ means that any participant designed with the protocol specified by T_1 in mind may also be used in a situation where the protocol specified by T_2 will be followed. Intuitively this pre-order between session types is generated by allowing more possibilities in branching types and restricting them in choice types. The reader is referred to Caires and Pfenning (2010), Gay and Hole (2005) and Honda *et al.* (1998) for more details on session types, including how they are associated with processes and what behaviour they guarantee.

Web services (Alonso *et al.* 2004; Bernardo *et al.* 2009) are distributed components which can be combined using standard communication protocols and machine-independent message formats to provide services to clients. To encourage reusability, descriptions of their behaviour are typically made available in searchable repositories (Oasis Standard

[†] This research was supported by SFI project 06 IN.1 1898.

2011). In papers such as Barbanera and de'Liguoro (2010), Carpineti *et al.* (2006), Castagna *et al.* (2009) and Laneve and Padovani (2007), a language of *contracts* has been proposed for describing behaviours; despite a very different surface syntax, the language of contracts is very similar in style to session types. In particular there is the sequencing of messages $\alpha_1.\alpha_2$, an external choice between behaviours $\sigma_1 + \sigma_2$ reminiscent of the branching type $\&\langle T_1, T_2 \rangle$, and an internal choice between allowed behaviours $\sigma_1 \oplus \sigma_2$, reminiscent of the choice type $\oplus\langle T_1, T_2 \rangle$.

The object of this paper is to study the precise relationship between these two formalisms. In particular for *first-order* session types, which do not allow the use of communication channels in messages, we show that the theory of session types, $\langle \mathcal{ST}, \preccurlyeq_{\mathrm{ST}} \rangle$, can be captured precisely using a natural pre-order over a subclass of contracts.

Contracts for web services serve two roles. A contract σ may describe the behaviour of a server offering some specific service. Dually a contract ρ may describe the behaviour expected of a client who wishes to avail of a particular service. Central to the theory of contracts for web services is the idea of compliance between such contracts, formalized as an asymmetric relation $\rho \dashv \sigma$; it has been defined in a variety of ways in papers such as Castagna *et al.* (2009) and Laneve and Padovani (2007, 2008). This leads to two natural pre-orders on contracts, defined set theoretically:

— the server pre-order: $\sigma_1 \sqsubseteq_{\mathrm{SRV}} \sigma_2$ if for every (client) contract ρ, $\rho \dashv \sigma_1$ implies $\rho \dashv \sigma_2$
— the client pre-order: $\rho_1 \sqsubseteq_{\mathrm{CLT}} \rho_2$ if for every (server) contract σ, $\rho_1 \dashv \sigma$ implies $\rho_2 \dashv \sigma$.

As we have already stated session types are more or less a syntactic variant of contracts; formally there is a straightforward translation $\mathcal{M}(T)$ of session types into contracts. Unfortunately neither of the relations $\sqsubseteq_{\mathrm{SRV}}, \sqsubseteq_{\mathrm{CLT}}$ are sound with respect to sub-typing; specifically there are session types T_1, T_2 such that $T_1 \preccurlyeq_{\mathrm{ST}} T_2$ but $\mathcal{M}(T_1)$ and $\mathcal{M}(T_2)$ are unrelated as contracts.

The problem lies in the fact that, viewed as constraints on behaviour, session types are much more constraining than contracts. We therefore isolate a subset of contracts, which we call *session contracts*, $\mathcal{SC}$, that are the range of the translation function $\mathcal{M}$. This enables us to define a sub-server relation and a sub-client relation, $\sqsubseteq^{\mathcal{SC}}_{\mathrm{SRV}}$ and $\sqsubseteq^{\mathcal{SC}}_{\mathrm{CLT}}$ respectively, on these contracts. Even though these relations are respectively coarser than $\sqsubseteq_{\mathrm{SRV}}$ and $\sqsubseteq_{\mathrm{CLT}}$, it turns out that these relations are still unsound with respect to session sub-typing. But in the main result of the paper we show that by combining these pre-orders we obtain *full abstraction*, that is a sound and complete model for session types.

Throughout the paper we follow the approach of Castagna *et al.* (2009) and Padovani (2010), and think of 'contracts' only as terms of (some dialect of) CCS without τ's (Nicola and Hennessy 1987); in particular, our notion of contract is completely independent of that discussed in McNeile (2010) and Meyer (1997).

1.1. *Contributions*

The contributions of this paper are the following: Theorem 5.7 provides what, to the best of our knowledge, is the first fully abstract model of session types in terms of contracts. Theorems 4.14 and 4.24 are the first alternative characterizations of the server

S, T	$::=$		**Session types**
		END	*satisfaction*
		$?[\mathtt{t}]; S$	*input*
		$![\mathtt{t}]; S$	*output*
		$\&\langle \mathtt{l}_1 \colon S_1, \dots, \mathtt{l}_n \colon S_n \rangle,\ n \geq 1$	*branch*
		$\oplus\langle \mathtt{l}_1 \colon S_1, \dots, \mathtt{l}_n \colon S_n \rangle,\ n \geq 1$	*choice*
		X	*type variable*
		$\mu X.\, S$	*recursion*

We impose the additional proviso that in a term the $\mathtt{l}_i$'s are pair-wise different.

Fig. 1. Session types (first-order).

and the client pre-orders on session contracts. Corollary 3.51 is the first published proof showing the equality between a first-order compliance-based refinement and a testing based must pre-order; moreover, the corollary means that the equality holds regardless of the co-action relation used to let contracts interact.

1.2. *Structure of the paper*

The paper is organized as follows. In the next section we give the definition of session types and the sub-typing between them; this material is taken directly from Gay and Hole (2005), although our definition is based on first-order types; however, we allow a primitive sub-typing relation between the basic types.

In the subsequent section we study contracts. We use the language for contracts of Padovani (2010), and we provide a formulation of the notion of compliance which differs from the one of Padovani (2010) in that (a) it is co-inductively defined, and (b) it is parametrized over the co-action relations $\bowtie$. Then, disregarding the parameter $\bowtie$, we provide a co-inductive characterization of the server pre-order on contracts, and we prove that it equals the must pre-order over contracts. As we reason up to $\bowtie$, the result that we obtain is more general than a similar result presented in Laneve and Padovani (2007). In Section 4 we focus on a subset of contracts called session contracts $\mathcal{SC}$, this time giving co-inductive characterizations to both the restricted server pre-order $\sqsubseteq^{\mathcal{SC}}_{\text{SRV}}$ and the restricted client pre-order $\sqsubseteq^{\mathcal{SC}}_{\text{CLT}}$ over them. Due to the very restricted nature of these contracts, these co-inductive characterizations are purely in terms of their syntax. In Section 5, we tackle the central question of the paper. Having defined the (obvious) translation of session types into session contracts, we explain why the two natural pre-orders $\sqsubseteq^{\mathcal{SC}}_{\text{SRV}}$ and $\sqsubseteq^{\mathcal{SC}}_{\text{CLT}}$ are unsound relative to the sub-typing on session types. Finally, we prove that when combined they provide a sound and complete model; the proof is greatly facilitated by their co-inductive characterizations. The paper concludes with a brief look at related work.

2. Session types

The syntax of terms for session types is given by the language $L_{\mathcal{ST}}$ in Figure 1. It presupposes a denumerable set of labels $\mathbb{L}$, ranged over by $\mathtt{l}$, and a set of basic or ground

$$\frac{}{T}\ T \neq \mu Z.\,S \qquad \frac{T\{{}^{\mu Y.\,T}/_{Y}\}}{\mu Y.\,T}$$

Fig. 2. Inference rules for $\downarrow_{\text{DPT}}$ on *closed* terms.

types `BT` types ranged over by `t`. We also use a denumerable set of variables *Vars*, ranged over by X, in order to express recursive types.

The use of variables leads to the usual notion of *free* and *bound* occurrences of variables in terms in the standard manner; we say that a term is *closed* if it contains no free variables. We also have the standard notion of *capture avoidance* substitution of terms for free variables. For the sake of clarity let us recall this definition: a substitution $\mathbf{s}$ is a mapping from the set *Vars* to the set of terms in $L_{\mathcal{ST}}$. Let

$$\mathbf{s} - X = \begin{cases} \mathbf{s} \setminus \{(X, \mathbf{s}(X))\} & \text{if } X \in \mathit{dom}(\mathbf{s}) \\ \mathbf{s} & \text{otherwise.} \end{cases}$$

Then the result of applying a substitution $\mathbf{s}$ to the term S is defined as follows:

$$S\mathbf{s} = \begin{cases} \text{END} & \text{if } S = \text{END} \\ \mathbf{s}(X) & \text{if } S = X, \text{ and } X \in \mathit{dom}(S) \\ X & \text{if } S = X, \text{ and } X \notin \mathit{dom}(S) \\ ![\mathtt{t}];(S'\mathbf{s}) & \text{if } S = ![\mathtt{t}];S' \\ ?[\mathtt{t}];(S'\mathbf{s}) & \text{if } S = ?[\mathtt{t}];S' \\ \&\langle\, \mathtt{l}_1 : (S_1\mathbf{s}), \ldots, \mathtt{l}_n : (S_n\mathbf{s})\,\rangle & \text{if } S = \&\langle\, \mathtt{l}_1 : S_1, \ldots, \mathtt{l}_n : S_n\,\rangle \\ \oplus\langle\, \mathtt{l}_1 : (S_1\mathbf{s}), \ldots, \mathtt{l}_n : (S_n\mathbf{s})\,\rangle & \text{if } S = \oplus\langle\, \mathtt{l}_1 : S_1, \ldots, \mathtt{l}_n : S_n\,\rangle \\ \mu X.\,(S'(\mathbf{s} - X)) & \text{if } S = \mu X.\,S'. \end{cases}$$

In the final clause, the application of $\mathbf{s} - X$ embodies the idea that in $\mu X.\,S'$ occurrences of X in the sub-term S' are bound and therefore substitutions have no effect on them.

It is easy to check that the effect of a substitution depends only on free variables; that is, $S\mathbf{s}_1 = S\mathbf{s}_2$ whenever $\mathbf{s}_1(X) = \mathbf{s}_2(X)$ for every free variable X occurring in S. We use the symbol $\{{}^{T}/_{X}\}$ to denote the singleton substitution $\{(X, T)\}$.

We will only use guarded recursion, which we now explain formally. Let $\downarrow_{\text{DPT}}$ be the least fixed point of the functional on closed terms defined by the inference rules in Figure 2.

Intuitively, $T\downarrow_{\text{DPT}}$ means that the free variables in T occur after a type constructor, which differs from μ. Now we say that a term T is *guarded* if every sub-term of T that has the form $\mu X.\,S$ satisfies $S\downarrow_{\text{DPT}}$. Finally, we use $\mathcal{ST}$ to denote the set of closed guarded terms, and we refer to the elements in $\mathcal{ST}$ as *session types.*

Example 2.1. The property $\downarrow_{\text{DPT}}$ and the property of being guarded are different. Consider the term $T = \&\langle\, \mathtt{l} : \mu X.\,X\,\rangle$; it is not a variable and the top-most constructor in it is not a recursion, therefore $T\downarrow_{\text{DPT}}$. A sub-term of T is $\mu X.\,X$ and clearly $\mu X.\,X\downarrow_{\text{DPT}}$ is false; therefore T is not guarded.

The advantage of only using guarded terms is that we can unfold types so as to obtain their top-most type constructor. To explain this formally first let us consider the function *depth* from terms to $\mathbb{N}^\infty$ (the set of natural numbers augmented by ∞). This is defined as the least such function which satisfies:

$$depth(S) = \begin{cases} 1 + depth(S' \left\{ {}^{S}/_{X} \right\}) & \text{if } S = \mu X.S', \\ 0 & \text{otherwise.} \end{cases}$$

Note that $depth(\mu X.X) = \infty$, but one can show that when applied to terms that satisfy the predicate $\downarrow_{\text{DPT}}$, one always obtains a natural number.

Lemma 2.2. If $S\downarrow_{\text{DPT}}$ then $depth(S) \in \mathbb{N}$.

Proof. The proof is by rule induction on the derivation of $S\downarrow_{\text{DPT}}$. If the axiom was used then thanks to the side condition $S \neq \mu X.S'$, and so $depth(S) = 0$ because of the definition of *depth*. If the other rule was used then $S = \mu X.S'$, and the hypothesis of the rule implies $S\left\{ {}^{S'}/_{X} \right\}\downarrow_{\text{DPT}}$. Then, by definition of *depth*, $depth(S) = 1 + depth(S\left\{ {}^{S'}/_{X} \right\})$, and by the inductive hypothesis $depth(S\left\{ {}^{S'}/_{X} \right\}) \in \mathbb{N}$; hence $depth(S) \in \mathbb{N}$. □

Proposition 2.3. The depth of any session type is finite.

Proof. Follows from the definition of $\mathcal{ST}$ and Lemma 2.2. □

This function *depth* will therefore provide a measure of session types over which we can perform induction.

Definition 2.4 (unfolding Gay and Hole (2005)).
For all $T \in \mathcal{ST}$, define $\text{UNFOLD}(T)$ as follows:

$$\text{UNFOLD}(T) = \begin{cases} \text{UNFOLD}(T' \left\{ {}^{\mu X.T}/_{X} \right\}) & \text{if } T = \mu X.T' \\ T & \text{otherwise.} \end{cases}$$

Lemma 2.5. For every $T \in \mathcal{ST}$, $\text{UNFOLD}(T)$ is a well-defined session type.

Proof. We have to show that $\text{UNFOLD}(T)$ is closed and guarded. The proof is by induction on $depth(P)$. It relies on the fact that each step of unfolding replaces one variable with a closed and guarded term; hence the overall unfolding is closed. □

Intuitively, $\text{UNFOLD}(T)$ unfolds top-level recursive definitions until a type constructor appears, which is not μ. This will be extremely useful in manipulating session types.

We conclude this sub-section by showing some typical examples of session types, which we recall from the literature.

Example 2.6 (math server, Gay and Hole (2005)).
Consider the session type

$$\begin{aligned} S_1 = \mu X.\&\langle\, & \texttt{plus} : ?[\texttt{Int}];?[\texttt{Int}];![\texttt{Int}];X, \\ & \texttt{eq} : ?[\texttt{Int}];?[\texttt{Int}];![\texttt{Bool}];\text{END}\,\rangle \end{aligned}$$

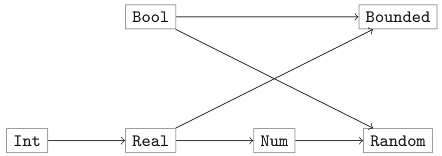

Fig. 3. A sub-type relation on a set of basic types BT; the arrow represents the relation $\preccurlyeq_{\mathtt{g}}$ (see Example 2.7).

This specifies the protocol of a server which offers two services at the labels `plus` and `eq`. The first expects the input of two integers, after which an integer is returned, and then the service is once more available. The second also expects two integers, then returns a boolean, after which the session terminates.

An extension to the service is specified by the type

$$\begin{aligned} S_2 = \mu X.\,\&\langle\, & \mathtt{plus}: ?[\mathtt{Int}]; ?[\mathtt{Int}]; ![\mathtt{Int}]; X, \\ & \mathtt{eq}: ?[\mathtt{Int}]; ?[\mathtt{Int}]; ![\mathtt{Bool}]; \mathrm{END}, \\ & \mathtt{neg}: ?[\mathtt{Bool}]; ![\mathtt{Bool}]; \mathrm{END}\,\rangle \end{aligned}$$

This provides in addition queries for negation.

2.1. *Sub-typing*

There are three sources for the sub-typing relation over types. The first is some predefined pre-order over the basic types, $\mathtt{t}_1 \preccurlyeq_{\mathtt{g}} \mathtt{t}_2$, which intuitively says that all data-values that have type $\mathtt{t}_1$ may be safely used where data-values of type $\mathtt{t}_2$ are expected.

Example 2.7. An example of sub-typing on base types is given in Figure 3, for the ensuing set of types, $\mathtt{BT} = \{\mathtt{Bounded}, \mathtt{Bool}, \mathtt{Int}, \mathtt{Real}, \mathtt{Num}, \mathtt{Random}\}$. In the figure, the pre-order $\preccurlyeq_{\mathtt{g}}$ is depicted by the arrows; for instance, the arrow from type `Int` to type `Real` means that $\mathtt{Int} \preccurlyeq_{\mathtt{g}} \mathtt{Real}$.

More generally, if $[\![\mathtt{t}]\!]$ denotes the set of values of the basic type $\mathtt{t}$ then we can define $\preccurlyeq_{\mathtt{g}}$ by letting $\mathtt{t}_1 \preccurlyeq_{\mathtt{g}} \mathtt{t}_2$ whenever $[\![\mathtt{t}_1]\!] \subseteq [\![\mathtt{t}_2]\!]$. The other sources are two constructs of the language: the *branch* construct allows sub-typing by extending the set of labels involved, while in the *choice* construct the set of labels may be restricted (see Examples 2.8 and 2.9 below). Moreover, we will have the standard co-variance/contra-variance of input/output types (Pierce and Sangiorgi 1996), extended to both the *branch* and *choice* constructs.

Example 2.8 (sub-typing on branch types). In this example, we explain how the sub-typing relates the branch types. Consider the type

$$\textsc{BarTender} = \&\langle\, \mathtt{espresso}: T_1\,\rangle$$

Intuitively, the BarTender offers only the label `espresso`, thus all the customers satisfied by BarTender, are satisfied by any other type that offers *at least* the label `espresso`. Let

$$\begin{aligned}\text{ItalianBarTender} = \&\langle\, & \texttt{espresso}: T_1',\\ & \texttt{deka}: T_2',\\ & \texttt{doubleespresso}: T_3' \,\rangle\end{aligned}$$

Following the intuition, The ItalianBarTender will satisfy all the customers satisfied by the BarTender; this is formalized by the sub-typing, which relates the two types as follows

$$\text{BarTender} \preccurlyeq_{\text{ST}} \text{ItalianBarTender}$$

as long as also the continuations T_1 and T_1' are related as well (i.e. $T_1 \preccurlyeq_{\text{ST}} T_1'$).

We have shown that, intuitively, it is safe to replace a branch type with a branch type that offers more labels.

Example 2.9 (sub-typing on choice types). In this example, we show how the sub-typing relate the choice types. Let ItalianCustomer describe the different coffees that a process may want to order when interacting with a bar tender.

$$\begin{aligned}\text{ItalianCustomer} = \oplus\,\langle\, & \texttt{espresso}: T_1',\\ & \texttt{deka}: T_2',\\ & \texttt{doubleespresso}: T_3' \,\rangle\end{aligned}$$

All the bar tenders that are able to satisfy this range of choices, have to offer *at least* the four labels that appear in ItalianCustomer. Now consider the type

$$\text{Customer} = \oplus\langle\, \texttt{espresso}: T_1' \,\rangle$$

Since Customer chooses among fewer options than ItalianCustomer, it is safe to use a channel at type Customer in place of a channel at type ItalianCustomer. This is formalized by the sub-typing relation as follows,

$$\text{ItalianCustomer} \preccurlyeq_{\text{ST}} \text{Customer}$$

In this example we have shown that, intuitively, it is safe to replace a choice type, with a choice type that chooses among fewer labels.

However, because of the recursive nature of our collection of types, the formal definition of the sub-typing relation is given co-inductively.

Definition 2.10 (type simulation). Let $\mathcal{P}(X)$ denote the powerset of a set X and let $\mathcal{F}_{\preccurlyeq_{\text{ST}}} : \mathcal{P}(\mathcal{ST}^2) \longrightarrow \mathcal{P}(\mathcal{ST}^2)$ be the function defined so that $(T, U) \in \mathcal{F}_{\preccurlyeq_{\text{ST}}}(\mathcal{R})$ whenever one of the following holds:

i. if $\textsc{unfold}(T) = \textsc{end}$ then $\textsc{unfold}(U) = \textsc{end}$
ii. if $\textsc{unfold}(T) = ?[\mathtt{t}_1]; S_1$ then $\textsc{unfold}(U) = ?[\mathtt{t}_2]; S_2$ and $(S_1, S_2) \in \mathcal{R}$ and $\mathtt{t}_1 \preccurlyeq_{\mathtt{g}} \mathtt{t}_2$
iii. if $\textsc{unfold}(T) = ![\mathtt{t}_1]; S_1$ then $\textsc{unfold}(U) = ![\mathtt{t}_2]; S_2$ and $(S_1, S_2) \in \mathcal{R}$ and $\mathtt{t}_2 \preccurlyeq_{\mathtt{g}} \mathtt{t}_1$
iv. if $\textsc{unfold}(T) = \&\langle\, \mathtt{l}_1 : S_1, \dots l_m : S_m \,\rangle$ then $\textsc{unfold}(U) = \&\langle\, \mathtt{l}_1 : S_1', \dots, \mathtt{l}_n : S_n' \,\rangle$ where $m \leqslant n$ and $(S_i, S_i') \in \mathcal{R}$ for all $i \in [1, \dots, m]$

v. if $\textsc{unfold}(T) = \oplus\langle \mathtt{l}_1 : S_1, \dots l_m : S_m \rangle$ then $\textsc{unfold}(U) = \oplus\langle \mathtt{l}_1 : S'_1, \dots, \mathtt{l}_n : S'_n \rangle$ where $n \leqslant m$ and $(S_i, S'_i) \in \mathcal{R}$ for all $i \in [1, \dots, n]$.

If $\mathcal{R} \subseteq \mathcal{F}_{\preccurlyeq_{\mathrm{ST}}}(\mathcal{R})$ then we say that $\mathcal{R}$ is a co-inductive type simulation. Let $\preccurlyeq_{\mathrm{ST}}$ denote the greatest solution of the equation $X = \mathcal{F}_{\preccurlyeq_{\mathrm{ST}}}(X)$; formally,

$$\preccurlyeq_{\mathrm{ST}} = \nu\mathcal{F}_{\preccurlyeq_{\mathrm{ST}}}.$$

We call $\preccurlyeq_{\mathrm{ST}}$ the *type simulation.* Standard arguments ensure that the relation $\preccurlyeq_{\mathrm{ST}}$ exists, that it is a typing relation, and indeed the greatest type simulation.

Example 2.11. Let T, S denote the types $\mu X.\, ?[\mathtt{Int}]; X$, $\mu X.\, ?[\mathtt{Real}]; X$ respectively. We can prove that $T \preccurlyeq_{\mathrm{ST}} S$, because the relation $\mathcal{R} = \{(T, S), (?[\mathtt{Int}]; T, ?[\mathtt{Real}]; S)\}$ is a type simulation, since $\mathtt{Int} \preccurlyeq_{\mathrm{g}} \mathtt{Real}$.

Referring to Example 2.6, one can also show that $S_1 \preccurlyeq_{\mathrm{ST}} S_2$ by providing an appropriate type simulation.

The requirement that session types be guarded is crucial for the sub-typing relation to be well defined. We explain this fact in the next example.

Example 2.12 (sub-typing and guardedness). Consider again the term $T = \&\langle \mathtt{l} : \mu X.\, X \rangle$ of Example 2.1. Suppose we wanted to check whether $T \preccurlyeq_{\mathrm{ST}} \&\langle \mathtt{l} : S \rangle$ for some session type S. The definition of $\preccurlyeq_{\mathrm{ST}}$ requires us to check whether

$$\textsc{unfold}(\mu X.\, X) \preccurlyeq_{\mathrm{ST}} \textsc{unfold}(S).$$

This check, though, cannot be done because $\textsc{unfold}(\mu X.\, X)$ is *not* defined at all (and the unfolding of S may not be defined either).

Proposition 2.13. The relation $\preccurlyeq_{\mathrm{ST}}$ is a pre-order on $\mathcal{ST}$.

Proof. See Gay and Hole (2005). □

In Gay and Hole (2005), the set of types $\mathcal{ST}$ are used to give a typing system for the pi calculus, and appropriate Type Safety and Type Preservation theorems are proved. Here instead our aim is to give a model to the set of types $\langle \mathcal{ST}, \preccurlyeq_{\mathrm{ST}} \rangle$ using contracts.

3. Contracts

We first define our language for contracts and give some examples. In the following sub-section, we define a natural server based pre-order on contracts, for which we give a behavioural co-inductive characterization. In the final sub-section, we investigate a closely related pre-order based on must testing.

3.1. *The contract language*

This sub-section is roughly divided in three parts. In the first one we define the language $L_{\mathcal{C}}$, and we are concerned with *syntactical* properties of its terms σ's; similarly to what we have done for session types, we introduce the unfolding of closed terms of L_c and a predicate $\downarrow_{\mathrm{DPT}}$ to guarantee that each contract can be unfolded (Lemma 3.1). Afterwards,

σ	$::=$		**Contracts**
		NIL	*termination*
		$\mathbf{1}$	*success*
		$\alpha.\sigma$	*action*
		$\sigma + \sigma$	*external choice*
		$\sigma \oplus \sigma$	*internal choice*
		x	*contract variable*
		$\mu x.\,\sigma$	*recursion*

Fig. 4. Contract grammar.

we give an operational semantics (Figure 5), and we discuss important *semantic* properties that we want contracts to enjoy; this will lead to the introduction of a predicate $\downarrow$, and two lemmas (Lemma 3.10 and 3.11) which ensure that (a) contracts do not diverge, and (b) silent moves lead to a finite number of derivatives. In the last part of the subsection, we describe how client–server interactions are modelled by contracts (Figure 7).

A language for contracts $L_{\mathcal{C}}$ is given in Figure 4. As with session types it uses a denumerable set of recursion variables *Vars*, here lower case, but also presupposes a set *Act* of actions, ranged over by α, which contracts can perform; as we will see the special action $\checkmark$, which we assume is *not* in *Act*, will be used to indicate the fulfilment of a contract. Intuitively the contract $\alpha.\sigma$ performs the action α and then behaves like σ; the sum $\sigma' + \sigma''$ is ready to behave either as σ' or as σ'' and the choice depends on the external environment. For this reason the operation $+$ is called *external* sum. The *internal* sum $\sigma' \oplus \sigma''$ represents a contract that can behave as σ' or as σ'', and the choice is taken by the contract independently from the environment. Such a decision can be due for instance to an IF statement in the process implementing the contracts. The symbol NIL denotes an empty contract, which intuitively can never be fulfilled, while **1** denotes the contract that is always satisfied.

Recursive definitions are handled in much the same way as session types and so we do not spell out the details; we assume a definition of capture-avoiding substitution **s**. Now we define the predicate $\downarrow_{\mathrm{DPT}}$, the function *depth*, and the function UNFOLD as in the previous section.

The function *depth* is the least one that satisfies

$$depth(\sigma) = \begin{cases} 1 + depth(\sigma'\{^{\sigma}/_{x}\}) & \text{if } \sigma = \mu x.\,\sigma', \\ 0 & \text{otherwise} \end{cases}$$

and the UNFOLD is the least function that satisfies:

$$\mathrm{UNFOLD}(\sigma) = \begin{cases} \mathrm{UNFOLD}(\sigma'\{^{\sigma}/_{x}\}) & \text{if } \sigma = \mu x.\,\sigma', \\ \sigma & \text{otherwise.} \end{cases}$$

These definitions are not arbitrary. As it happens, they let us prove Lemma 5.2, which, to our aim, is paramount (see Section 5).

Let $\downarrow_{\mathrm{DPT}}$ be the least fixed point of the rule functional defined on closed terms of $L_{\mathcal{C}}$ by the rules in Figure 2.

Similarly to what done in the previous section one can prove the following lemma.

$$\frac{}{\mathbf{1} \xrightarrow{\checkmark} \text{NIL}} \text{[A-OK]}$$

$$\frac{}{\alpha.\sigma \xrightarrow{\alpha} \sigma} \text{[A-PRE]} \qquad \frac{}{\mu x.\,\sigma \xrightarrow{\tau} \sigma\{^{\mu x.\,\sigma}/_x\}} \text{[A-UNF]}$$

$$\frac{}{\sigma \oplus \rho \xrightarrow{\tau} \sigma} \text{[A-IN-L]} \qquad \frac{}{\sigma \oplus \rho \xrightarrow{\tau} \rho} \text{[A-IN-R]}$$

$$\frac{\sigma \xrightarrow{\lambda} \sigma'}{\sigma + \rho \xrightarrow{\lambda} \sigma'} \text{[R-EXT-L]} \qquad \frac{\rho \xrightarrow{\lambda} \rho'}{\sigma + \rho \xrightarrow{\lambda} \rho'} \text{[R-EXT-R]}$$

$$\frac{\sigma \xrightarrow{\tau} \sigma'}{\sigma + \rho \xrightarrow{\tau} \sigma' + \rho} \text{[R-INT-L]} \qquad \frac{\rho \xrightarrow{\tau} \rho'}{\sigma + \rho \xrightarrow{\tau} \sigma + \rho'} \text{[R-INT-R]}$$

Fig. 5. Inference rules for the semantics of closed terms of $L_{\mathcal{C}}$, where $\lambda \in Act_{\checkmark}$.

Lemma 3.1. Let $\sigma\downarrow_{\text{DPT}}$. Then

i. the depth of σ is finite: $depth(\sigma) \in \mathbb{N}$
ii. the term $\text{UNFOLD}(\sigma)$ is defined and if σ is closed then $\text{UNFOLD}(\sigma)$ is closed.

Proof. The proof of part ii relies on (i) and is similar to the proof of Lemma 2.2. □

An operational semantics for the closed terms of the language $L_{\mathcal{C}}$ is given in Figure 5. The judgements are of the form

$$\sigma \xrightarrow{\mu} \sigma', \qquad \mu \in Act_{\tau\checkmark}$$

where we use $Act_{\tau\checkmark}$ as a shorthand for the set $Act \cup \{\tau, \checkmark\}$, and $Act_{\checkmark}$ as shorthand for $Act \cup \{\checkmark\}$. The judgement $\sigma \xrightarrow{\alpha} \sigma$, where $\alpha \in Act$ has the obvious meaning; $\sigma \xrightarrow{\tau} \sigma'$, means that the contract σ is resolved to the contract σ' by some internal computation, while $\sigma \xrightarrow{\checkmark} \sigma'$ represents the reporting of the successful completion of a computation.

Let $\xrightarrow{\tau}{}^{*}$ denote the reflexive transitive closure of $\xrightarrow{\tau}$.

We are now ready to show two properties of UNFOLD. We will use them in the rest of the paper.

Lemma 3.2. Let σ be a contract.

i. If $\sigma \stackrel{\tau}{\not\longrightarrow}$ then $\text{UNFOLD}(\sigma) = \sigma$.
ii. $\sigma \xrightarrow{\tau}{}^{*} \text{UNFOLD}(\sigma)$.

Proof. Part i is proved by structural induction on σ. We prove part ii; the argument is by induction on $depth(\sigma)$. If $depth(\sigma) = 0$ then from the definition of *depth* it follows that $\sigma \neq \mu x.\sigma'$; by definition of UNFOLD then $\text{UNFOLD}(\sigma) = \sigma$. The reflexivity of $\xrightarrow{\tau}{}^{*}$ implies $\sigma \xrightarrow{\tau}{}^{*} \text{UNFOLD}(\sigma)$.

If $depth(\sigma) > 1$ then, due to the definition of *depth*, $\sigma = \mu x.\sigma'$. The definition of UNFOLD implies that $\text{UNFOLD}(\sigma) = \text{UNFOLD}(\sigma'\{^{\sigma}/_x\})$, while the definition of *depth* implies $depth(\sigma) = 1 + depth(\sigma'\{^{\sigma}/_x\})$, and therefore $depth(\sigma'\{^{\sigma}/_x\})$ is smaller than

$depth(\sigma)$. We are now allowed to use the inductive hypothesis on $\sigma'\{^{\sigma}/_{x}\}$:

$$\sigma'\{^{\sigma}/_{x}\} \xrightarrow{\tau}^{*} \text{UNFOLD}(\sigma'\{^{\sigma}/_{x}\}).$$

We use rule [A-UNF] (see Figure 5) to infer $\sigma \xrightarrow{\tau} \sigma'\{^{\sigma}/_{x}\}$, and then the transitivity of the relation $\xrightarrow{\tau}^{*}$ to obtain

$$\sigma \xrightarrow{\tau}^{*} \text{UNFOLD}(\sigma'\{^{\sigma}/_{x}\}).$$

We already know that $\text{UNFOLD}(\sigma) = \text{UNFOLD}(\sigma'\{^{\sigma}/_{x}\})$, and, by applying this equality to the reduction sequence above, we get

$$\sigma \xrightarrow{\tau}^{*} \text{UNFOLD}(\sigma).$$

This concludes the proof. □

Let

$$S(\sigma) = \{\, \sigma' \mid \sigma \xrightarrow{\mu} \sigma' \text{ for some } \mu \in Act_{\tau\checkmark} \,\}$$
$$F(\sigma) = \{\, \sigma' \mid \sigma \xrightarrow{\tau}^{*} \sigma' \,\}.$$

One might think that $F(\sigma)$ is finite if and only if σ does not diverge. This is not the case.

Example 3.3 (divergence and finite derivatives). Consider the terms $\mu x.\, x$ and $\mu x.(\,\text{NIL} \oplus x\,)$. Both terms *diverge*, in the sense that they perform an infinite sequence of τ's:

$$\mu x.\, x \xrightarrow{\tau} \mu x.\, x, \quad \mu x.(\,\text{NIL} \oplus x\,) \xrightarrow{\tau} \text{NIL} \oplus \mu x.(\,\text{NIL} \oplus x\,) \xrightarrow{\tau} \mu x.(\,\text{NIL} \oplus x\,).$$

On the other hand we have

$$F(\mu x.\, x) = S(\mu x.\, x) = \{\, \mu x.\, x \,\}$$

and

$$\begin{aligned} F(\mu x.(\,\text{NIL} \oplus x\,)) &= \{\, \mu x.(\,\text{NIL} \oplus x\,), \text{NIL} \oplus \mu x.(\,\text{NIL} \oplus x\,) \,\} \\ S(\mu x.(\,\text{NIL} \oplus x\,)) &= \{\, \text{NIL} \oplus \mu x.(\,\text{NIL} \oplus x\,) \,\}. \end{aligned}$$

The set $F(\sigma)$ is not finite for every σ.

Example 3.4 (infinite derivatives).
We show two terms σ, σ' such that $F(\sigma)$ and $F(\sigma')$ are infinite. Let $\sigma = \mu x.(\,\alpha.x + x\,)$ and $\sigma' = \mu x.(\,\alpha.\text{NIL} + (x \oplus x)\,)$. According to the rules in Figure 5 one can infer:

$$\sigma \xrightarrow{\tau} (\alpha.\sigma + \sigma) \xrightarrow{\tau} (\alpha.\sigma + (\alpha.\sigma + \sigma)) \xrightarrow{\tau} \ldots$$

and

$$\sigma' \xrightarrow{\tau} \alpha.\text{NIL} + (\sigma' \oplus \sigma') \xrightarrow{\tau} \alpha.\text{NIL} + \sigma' \xrightarrow{\tau} \alpha.\text{NIL} + (\alpha.\text{NIL} + (\sigma' \oplus \sigma')) \xrightarrow{\tau} \ldots$$

The root of the problem is that the inference rules [R-INT-L] and [R-INT-R] do not resolve the external sum.

On the other hand the finiteness of $S(\sigma)$ is easy to prove.

$$\overline{\mathbf{1}} \qquad \overline{\text{NIL}} \qquad \overline{\alpha.\sigma} \qquad \frac{\rho \quad \sigma}{(\sigma \oplus \rho)} \qquad \frac{\rho \quad \sigma}{(\rho + \sigma)} \qquad \frac{\sigma\{^{\mu x.\sigma}/_x\}}{\mu x.\sigma}$$

Fig. 6. Inference rules for $\downarrow$ over closed terms of $L_{\mathcal{C}}$.

Lemma 3.5 (finite branches). The set $S(\sigma)$ is finite for every σ.

Proof. The proof is by structural induction. If $\sigma = \text{NIL}$ then plainly $S(\sigma) = \emptyset$. Otherwise we proceed as follows,

- — if $\sigma = \alpha.\sigma'$ then the only applicable rule (see Figure 5) is [A-PRE], and so $S(\sigma) = \{\sigma'\}$;
- — if $\sigma = \mu x.\sigma'$ then only rule [A-UNF] can be applied, so $S(\sigma) = \{\sigma'\{^{\sigma}/_x\}\}$;
- — if $\sigma = \sigma' + \sigma''$ then σ' and σ'' are both smaller than σ. The inductive hypothesis tells us that $S(\sigma')$ and $S(\sigma'')$ are finite. Rules [R-EXT-L], [R-EXT-R], [R-INT-L] and [R-INT-R] in Figure 5 ensure that $S(\sigma) = S(\sigma') \cup S(\sigma'')$. This implies that the cardinality of $S(\sigma)$ is finite;
- — if $\sigma = \sigma' \oplus \sigma''$ the argument is alike the previous one.

$\square$

Example 3.6 (divergence and $\downarrow_{\text{dpt}}$). Consider the term

$$\sigma = \mu x.(\, x \oplus x\,).$$

Using the rules in Figure 2 one can prove that $\sigma\downarrow_{\text{DPT}}$; consequently $depth(\sigma)$ is finite and $\text{UNFOLD}(\sigma)$ is well defined; in particular $depth(\sigma) = 1$ and $\text{UNFOLD}(\sigma) = \sigma \oplus \sigma$. Note now that the term σ engages in an infinite sequence of internal moves

$$\sigma \xrightarrow{\tau} \sigma \oplus \sigma \xrightarrow{\tau} \sigma \xrightarrow{\tau} \sigma \oplus \sigma \xrightarrow{\tau} \ldots$$

In other words the term σ diverges. Similarly, one can reason that terms as $\mu x.(\,\text{NIL} \oplus x\,)$ and $\mu x.(\,\alpha \oplus x\,)$ suffer the same issue.

Throughout the paper, we want to deal only with terms that do *not* diverge and with finite $F(\sigma)$. To isolate the σ's that converge we use the predicate $\downarrow$. Formally, we define it as the least fixed point of the functional given by the inference rules in Figure 6. The predicate $\downarrow$ is essentially a strengthened version of $\downarrow_{\text{DPT}}$.

Proposition 3.7. For every $\sigma \in L_{\mathcal{C}}$, $\sigma\downarrow$ implies $\sigma\downarrow_{\text{DPT}}$.

Proof. Straightforward from the definitions of the predicates. $\square$

The predicate $\downarrow$ is preserved by silent moves.

Lemma 3.8. Let $\sigma\downarrow$. If $\sigma \xrightarrow{\tau} \sigma'$ then $\sigma'\downarrow$.

Proof. The proof proceeds by rule induction on the derivation of $\sigma \xrightarrow{\tau} \sigma'$.

The only interesting case is when the silent move $\sigma \xrightarrow{\tau} \sigma'$ is inferred by using rule [R-INT-L] or rule [R-INT-R] (see Figure 5). Suppose rule [R-INT-L] was used. Then $\sigma = \sigma_1 + \sigma_2$, $\sigma' = \sigma_1' + \sigma_2$, and the derivation is

$$\frac{\sigma_1 \xrightarrow{\tau} \sigma_1'}{\sigma_1 + \sigma_2 \xrightarrow{\tau} \sigma_1' + \sigma_2}\ .$$

We have to prove that $\sigma_1' + \sigma_2\!\downarrow$; the definition of $\downarrow$ ensures that, to this aim, it is enough to show that (a) $\sigma_1'\!\downarrow$ and that (b) $\sigma_2\!\downarrow$ (see Figure 6). Point (b) follows from the hypothesis: since $\sigma\!\downarrow$, the equality $\sigma = \sigma_1 + \sigma_2$ implies that $\sigma_1\!\downarrow$ and that $\sigma_2\!\downarrow$. The last fact is exactly point (b).

Now, by using the fact that $\sigma_1\!\downarrow$, we prove point (a). The derivation shown above let us use rule induction; the lemma holds for σ_1: if $\sigma_1\!\downarrow$ and $\sigma_1 \xrightarrow{\tau} \hat{\sigma}_1$ then $\hat{\sigma}'_1\!\downarrow$. We know that $\sigma_1\!\downarrow$ and that $\sigma_1 \xrightarrow{\tau} \sigma_1'$, thus it must be $\sigma_1'\!\downarrow$.

If rule [r-Int-R] was used the argument is similar. □

We have also the converse.

Lemma 3.9. Let σ be a closed term of L_C. Then $\sigma\!\downarrow$ if and only if $\sigma \xrightarrow{\tau} \sigma'$ implies $\sigma'\!\downarrow$.

Proof. The *only if* side of the lemma is Lemma 3.8, so we are required to prove only that if $\sigma \xrightarrow{\tau} \sigma'$ implies $\sigma'\!\downarrow$, then $\sigma\!\downarrow$.

Let σ be a closed term of L_C such that $\sigma \xrightarrow{\tau} \sigma'$ implies $\sigma'\!\downarrow$. We have to show that $\sigma\!\downarrow$.

The proof is by structural induction on the form of σ; the only interesting case is when the term σ is an external sum. In that case, $\sigma = \sigma_1 + \sigma_2$, so to prove that $\sigma_1\!\downarrow$ it is enough to show that $\sigma_1\!\downarrow$ and that $\sigma_2\!\downarrow$.

We prove $\sigma_1\!\downarrow$. The term σ_1 is a sub-term of σ, hence structural induction guarantees that the lemma holds for σ_1: if $\sigma_1 \xrightarrow{\tau} \sigma_1'$ implies $\sigma_1'\!\downarrow$, then $\sigma_1\!\downarrow$. Assume that $\sigma_1 \xrightarrow{\tau} \sigma_1'$; thanks to the structure of σ we can derive

$$\frac{\sigma_1 \xrightarrow{\tau} \sigma_1'}{\sigma_1 + \sigma_2 \xrightarrow{\tau} \sigma_1' + \sigma_2}\,[\text{r-Int-L}]\ .$$

Now the hypothesis of the lemma implies that $\sigma_1' + \sigma_2\!\downarrow$, so from the definition of $\downarrow$ it follows that $\sigma_1'\!\downarrow$ and that $\sigma_2\!\downarrow$.

We have shown that $\sigma_1 \xrightarrow{\tau} \sigma_1'$ implies $\sigma_1'\!\downarrow$, so from the inductive hypothesis it follows that $\sigma_1\!\downarrow$. Moreover, we have also shown that $\sigma_2\!\downarrow$; we have proven that $\sigma\!\downarrow$.

The argument for rule [r-Int-L] is analogous, and left to the reader. □

The predicate $\downarrow$ let us give an inductive characterization of the convergent terms.

Lemma 3.10 (convergence). Let σ be a closed term of the set L_C; $\sigma\!\downarrow$ if and only if there exists a natural number k such that $\sigma \xrightarrow{\tau}{}^{n} \sigma'$ implies $n \leqslant k$.

Proof. The *only if* side is by rule induction on why $\sigma\!\downarrow$. We prove the *if* side, which states that if there exists a $k \in \mathbb{N}$ such that $\sigma \xrightarrow{\tau}{}^{n} \sigma'$ implies $n \leqslant k$, then $\sigma\!\downarrow$.

The argument is an induction on k. If $k = 0$ then σ cannot perform τ, and so it is either **1**, nil or a prefix $\alpha.\sigma'$. In all these cases we can easily infer $\sigma\!\downarrow$.

If $k > 0$ then σ performs a τ, so suppose $\sigma \xrightarrow{\tau} \sigma'$; it follows that $\sigma' \xrightarrow{\tau}{}^{m} \sigma''$ implies $m \leqslant k - 1$. This means that there exists a k' such that $\sigma' \xrightarrow{\tau}{}^{m} \sigma''$ implies $m \leqslant k'$. Since

$k-1<k$, we can apply the inductive hypothesis to σ'; from this application it follows that $\sigma'\downarrow$.

As yet, we have proven that $\sigma \xrightarrow{\tau} \sigma'$ implies $\sigma'\downarrow$; this allows us to apply Lemma 3.9, which ensures that $\sigma\downarrow$. □

It is easy to see that a converging term σ has finite $F(\sigma)$.

Lemma 3.11. For every $\sigma \in L_{\mathcal{C}}$ if $\sigma\downarrow$ then $F(\sigma)$ is finite.

Proof. We can prove it by rule induction on the derivation of $\sigma\downarrow$. □

We are ready to define the set of contracts.

Definition 3.12 (contracts). Let $\mathcal{C}$ denote the set of all terms σ of $L_{\mathcal{C}}$ which are closed and such that if $\sigma \xrightarrow{s}{}^{*} \sigma'$ for some $s \in Act^{\star}$, then $\sigma\downarrow$. We refer to these terms as *contracts*.

Proposition 3.13. Let σ be a contract, then $\textsc{unfold}(\sigma)$ is a well-defined contract.

Proof. We have to show that $\textsc{unfold}(\sigma)$ is closed, and that it satisfies $\downarrow$. The first fact is part ii of Lemma 3.1. The second fact follows from part (ii) of Lemmas 3.2 and 3.8. □

Example 3.14 (e-vote, (Barbanera and de'Liguoro 2010; Laneve and Padovani 2008)).

$$\begin{aligned}
\text{Ballot} &= \mu x.\ ?\texttt{Login}.(\ !\texttt{Wrong}.x \oplus !\texttt{Ok}.(\ ?\texttt{VoteA}.x + ?\texttt{VoteB}.x\ )\ )\\
\text{Voter} &= \mu x.\ !\texttt{Login}.(\ ?\texttt{Wrong}.x + ?\texttt{Ok}.(\ !\texttt{VoteA}.\mathbf{1} \oplus !\texttt{VoteB}.\mathbf{1}\ )\ )
\end{aligned}$$

A process offering contract Ballot implements a service for e-voting. Such a service lets a client log in. If the log in fails the services starts anew, while if the log in succeeds the two actions are offered to the environment, namely `VoteA` and `VoteB`.

The contract Voter is a recursive client for the protocol described by the contract Ballot.

Example 3.15 (e-commerce, Bernardi *et al.* (2008)).

$$\begin{aligned}
\text{Customer} &= !\texttt{Request}.(\ !\texttt{PayDebit}.\rho' \oplus\\
&\qquad\qquad\quad !\texttt{PayCredit}.\rho' \oplus\\
&\qquad\qquad\quad !\texttt{PayCash}.\mathbf{1}\ )\\
\rho' &= !\texttt{Long}.?\texttt{Bool}.\mathbf{1}\\
\text{Bank} &= \mu x.\ ?\texttt{Request}.(\ ?\texttt{PayCredit}.?\texttt{Long}.!\texttt{Bool}.x +\\
&\qquad\qquad\qquad\quad ?\texttt{PayDebit}.?\texttt{Long}.!\texttt{Bool}.x +\\
&\qquad\qquad\qquad\quad ?\texttt{PayCash}.x\ )
\end{aligned}$$

The contracts above describe the conversation that should take place between a client (which offers the contract Customer) and a bank (which has contract Bank) involved in an on-line payment. The conversation unfolds as follows: the Customer sends a request to the bank and afterwards it chooses the payment method; the choice is taken by an internal sum and this means that the decision of the Customer is independent from the environment (i.e., the Bank contract). If the Customer decides to pay by cash then no other action has to be taken; while if the payment is done by debit or credit card the Customer has to send the card number, this is represented by the output !Long. After the card number has been received the Bank answers with a boolean. Intuitively, this represents the fact that the bank can approve or reject the payment. The Customer protocol finishes after such boolean has been received, while the Bank starts anew.

3.1.1. *Client–Server interactions and the compliance relation.* Contracts are expressive enough to encode XML based languages such as WS-BPEL activities and WSCL diagrams (Castagna *et al.* 2009); moreover in Carpineti *et al.* (2006) it is shown how to assign contracts to a subset of CCS processes. Intuitively, if a process, such as a server, is assigned a contract σ then it guarantees to support the behaviour described in σ. The interaction between servers and clients can be described at the level of their contracts, by defining a binary operation $\rho \parallel \sigma$ between their contracts and describing the evolution of the contracts as they interact. This interacting semantics is given in Figure 7, where the judgements are of the form $\rho \parallel \sigma \xrightarrow{\tau} \rho' \parallel \sigma'$. It presupposes a binary relation $\bowtie$ on *Act*, where $\alpha \bowtie \beta$ means that the action α can synchronize with the action β. This relation can be instantiated in various ways depending on the particular set of actions *Act*.

Example 3.16. Suppose we take *Act* to be $\{a?, a! \mid a \in A\}$, where A is a set of communicating channels. Then define

$$\alpha \bowtie_i \beta \qquad \text{whenever } \alpha = a?, \beta = a! \text{ or } \alpha = a!, \beta = a? \text{ for some } a \in A.$$

The relation $\bowtie_i$ represents synchronization on channels.

We will use a more elaborate set of actions when interpreting session types as contracts. Recall from Section 2 the set of basic types $\mathtt{BT}$, the set of labels $\mathbb{L}$ used in session types, and Example 2.7. We can define *Act* to be the set

$$\{\, ?\mathtt{b},\ !\mathtt{b} \mid \mathtt{b} \in \mathtt{BT} \,\} \cup \{\, \mathtt{l}?,\ \mathtt{l}! \mid \mathtt{l} \in \mathbb{L} \,\}$$

with $\bowtie_c$ determined by

$$\alpha \bowtie_c \beta \qquad \text{whenever} \begin{cases} \alpha = ?\mathtt{b},\ \beta = !\mathtt{b}' & \mathtt{b}' \preccurlyeq_g \mathtt{b} \\ \alpha = !\mathtt{b},\ \beta = ?\mathtt{b}' & \mathtt{b} \preccurlyeq_g \mathtt{b}' \\ \alpha = ?\mathtt{l},\ \beta = !\mathtt{l} & \\ \alpha = !\mathtt{l},\ \beta = ?\mathtt{l} & . \end{cases}$$

Using the basic sub-typing relation depicted in Figure 3 the following examples should be clear:

i. $?\mathtt{Num} \bowtie_c !\mathtt{Int}$: a contract that can read a datum of type $\mathtt{Num}$ can read a datum of type $\mathtt{Int}$ because $\mathtt{Int} \preccurlyeq_g \mathtt{Num}$.
ii. $?\mathtt{Int} \not\bowtie_c !\mathtt{Num}$: conversely a contract ready to read a datum of type $\mathtt{Int}$ cannot read a datum of type $\mathtt{Num}$ because $\mathtt{Num} \not\preccurlyeq_g \mathtt{Int}$.
iii. $?\mathtt{Random} \bowtie_c !\mathtt{Bool}$: as in point (i), $\mathtt{Bool} \preccurlyeq_g \mathtt{Random}$ hence an interaction between the actions $?\mathtt{Random}$ and $!\mathtt{Bool}$ is safe.

$$\frac{\rho \xrightarrow{\tau} \rho'}{\rho \parallel \sigma \xrightarrow{\tau} \rho' \parallel \sigma} \text{[P-SIL-L]} \qquad \frac{\sigma \xrightarrow{\tau} \sigma'}{\rho \parallel \sigma \xrightarrow{\tau} \rho \parallel \sigma'} \text{[P-SIL-R]}$$

$$\frac{\rho \xrightarrow{\alpha} \rho' \quad \sigma \xrightarrow{\beta} \sigma'}{\rho \parallel \sigma \xrightarrow{\tau} \rho' \parallel \sigma'} \; \alpha \bowtie \beta \text{ [P-SYNCH]}$$

Fig. 7. Inference rules for contract interaction.

Having described how interactions between clients and servers affect their contracts, let us describe, by means of a relation, when a client (guaranteeing a) contract ρ can safely interact with a server (guaranteeing a) contract σ. Indeed, we shall formalize the meaning of 'safely'.

The central notion is that of *compliance* between contracts. This is defined co-inductively and uses the predicate on contracts $\rho \xrightarrow{\checkmark}$ which intuitively means that the contract ρ has already been satisfied. Our definition is a variation on that of *compliance* in Laneve and Padovani (2007, 2008) and Padovani (2010).

Definition 3.17 (compliance relation).
Let $\mathcal{F}_{\dashv} : \mathcal{P}(\mathcal{C}^2) \longrightarrow \mathcal{P}(\mathcal{C}^2)$ be the function defined so that $(\rho, \sigma) \in \mathcal{F}_{\dashv}(\mathcal{R})$ whenever both the following hold:

i. if $\rho \parallel \sigma \overset{\tau}{\not\longrightarrow}$ then $\rho \xrightarrow{\checkmark}$
ii. if $\rho \parallel \sigma \xrightarrow{\tau} \rho' \parallel \sigma'$ then $(\rho', \sigma') \in \mathcal{R}$.

If $\mathcal{R} \subseteq \mathcal{F}_{\dashv}(\mathcal{R})$ then we say that $\mathcal{R}$ is a co-inductive *compliance* relation. Let $\dashv$ denote the greatest solution of the equation $X = \mathcal{F}_{\dashv}(X)$; formally,

$$\dashv = \nu \mathcal{F}_{\dashv}$$

We call $\dashv$ the *compliance* relation. If $\rho \dashv \sigma$ we say that the contract ρ *complies with* the contract σ.

Notice that there is an asymmetry in the relation $\rho \dashv \sigma$; the intention is that any client running contract ρ when interacting with a server running contract σ will be satisfied, in the sense that either the interaction between client and server will go on indefinitely, or, if the interaction gets stuck, the client will end on its own in a state in which it is satisfied, $\rho \xrightarrow{\checkmark}$.

Example 3.18 (compliance and divergent terms).
In order that the relation $\dashv$ captures the intuition described above, it is crucial that $\mathcal{C}$ contains no divergent terms. Had we admitted them, then for every ρ the relation

$$\{\, (\rho', \; \mu x. \, x) \mid \rho \xrightarrow{\tau}{}^{*} \rho' \,\}$$

would have been a perfectly fine co-inductive compliance. Note, though, that the client contract ρ is by no means satisfied by the server.

Example 3.19 (compliance and $\checkmark$). According to our definition of compliance, the client need not ever perform $\checkmark$. For example, suppose $\alpha \bowtie \beta$ and consider the ensuing set

$$\{(\,\mu x.\,\alpha.x,\ \mu y.\,\beta.y\,)\}.$$

This set is a co-inductive compliance relation, and the client contract, $\mu x.\,\alpha.x$, does not perform $\checkmark$ at all.

Example 3.20. The fact that NIL cannot be satisfied is formally expressed by the fact that

$$\text{NIL} \not\dashv \sigma$$

for every contract σ. On the other hand $\mathbf{1}$ is always satisfied because we can prove that for every contract σ we have the following,

$$\mathbf{1} \dashv \sigma.$$

Suppose σ is a contract which cannot interact with the action α; by this we mean that $\sigma \xrightarrow{\tau}{}^{*} \xrightarrow{\beta}$ implies $\beta \not\bowtie \alpha$. Then

$$\mathbf{1} + \alpha.\rho \dashv \sigma$$

for every ρ, because ρ is guarded by an action that can never take place.

Referring to Example 3.14, it is routine work to check that the following relation is a co-inductive compliance.

$$\begin{aligned}
\mathcal{R} = \{&(\text{Voter}, \text{Ballot}),\\
&(\,?\texttt{Wrong}.\text{Voter} + ?\texttt{Ok}.(\,!\texttt{VoteA}.\mathbf{1} \oplus !\texttt{VoteB}.\mathbf{1}\,),\\
&!\texttt{Wrong}.\text{Ballot} \oplus !\texttt{Ok}.(\,?\texttt{VoteA}.\text{Ballot} + ?\texttt{VoteB}.\text{Ballot})),\\
&(\,?\texttt{Ok}.(\,!\texttt{VoteA}.\mathbf{1} \oplus !\texttt{VoteB}.\mathbf{1}\,), !\texttt{Ok}.(\,?\texttt{VoteA}.\text{Ballot} + ?\texttt{VoteB}.\text{Ballot}),\\
&(\,!\texttt{VoteA}.\mathbf{1} \oplus !\texttt{VoteB}.\mathbf{1}, ?\texttt{VoteA}.\text{Ballot} + ?\texttt{VoteB}.\text{Ballot}),\\
&(\,!\texttt{VoteA}.\mathbf{1}, ?\texttt{VoteA}.\text{Ballot} + ?\texttt{VoteB}.\text{Ballot}),\\
&(\,!\texttt{VoteB}.\mathbf{1}, ?\texttt{VoteA}.\text{Ballot} + ?\texttt{VoteB}.\text{Ballot}),\\
&(\mathbf{1}, \text{Ballot})\}
\end{aligned}$$

The previous example shows that on the client-side, the contracts $\mathbf{1}$ and NIL have opposite meanings, as the former is always satisfied, while the latter is never satisfied; thus a client whose contract is $\mathbf{1}$ is not equivalent to a client whose contract is NIL. On the server-side the situation is different; a server with contract $\mathbf{1}$ is equivalent to a server with contract NIL. We prove this fact.

Proposition 3.21. For every contract ρ, $\rho \dashv \mathbf{1}$ if and only if $\rho \dashv \text{NIL}$.

Proof. Suppose $\rho \dashv \mathbf{1}$; this means that there exists a co-inductive compliance $\mathcal{R}$ that contains $(\rho, \mathbf{1})$. Since $\mathbf{1}$ offers no interaction, the contract ρ enjoys the two properties which follow,

i. if $\rho \stackrel{\tau}{\not\longrightarrow}$ then $\rho \xrightarrow{\checkmark}$
ii. if $\rho \xrightarrow{\tau} \rho'$ then $(\rho', \mathbf{1}) \in \mathcal{R}$.

Knowing i, it is straightforward to show that

$$\mathcal{R}' = \{ (\rho', \textsc{nil}) \mid \rho \xrightarrow{\tau}{}^{*} \rho', \rho \dashv \mathbf{1} \}$$

is a co-inductive compliance. A symmetrical argument can be used to show also that

$$\mathcal{R}' = \{ (\rho', \mathbf{1}) \mid \rho \xrightarrow{\tau}{}^{*} \rho', \rho \dashv \textsc{nil} \}$$

is a co-inductive compliance. □

The following properties of the compliance relation will be useful later in the paper.

Lemma 3.22. Let ρ, σ_1, and σ_2 be contracts. The following hold:

i. if $\rho \dashv \sigma_1$, $\rho \dashv \sigma_2$ then $\rho \dashv \sigma_1 \oplus \sigma_2$
ii. if $\rho_1 \dashv \sigma$, $\rho_2 \dashv \sigma$ then $\rho_1 \oplus \rho_2 \dashv \sigma$.

Proof. As an example we outline the proof of (i). Let $\mathcal{R}$ be the relation defined by

$$\mathcal{R} = \{ (\rho, \sigma) \mid \rho \dashv \sigma \text{ or } \sigma = \sigma_1 \oplus \sigma_2 \text{ where } \rho \dashv \sigma_1 \text{ and } \rho \dashv \sigma_2 \}.$$

It is straightforward to show that $\mathcal{R}$ is a compliance relation, from which the result follows. □

Proposition 3.23. For all contracts ρ, σ, we have the following

a. if $\rho \dashv \sigma$ then $\rho \dashv \textsc{unfold}(\sigma)$
b. if $\rho \dashv \sigma$ then $\textsc{unfold}(\rho) \dashv \sigma$.

Proof. Both follow in a straightforward manner from part (ii) of Lemma 3.2 and part ii of Definition 3.17. □

The converse is also true:

Proposition 3.24. For all contracts ρ, σ, we have the following

a. if $\rho \dashv \textsc{unfold}(\sigma)$ then $\rho \dashv \sigma$
b. if $\textsc{unfold}(\rho) \dashv \sigma$ then $\rho \dashv \sigma$.

Proof. Let us look at the proof of (a). Let

$$\mathcal{R} = \{ (\rho, \sigma) \mid \rho \dashv \sigma \text{ or } \rho \dashv \textsc{unfold}(\sigma) \}.$$

The result will follow if we can prove that $\mathcal{R}$ is a co-inductive compliance relation, as given in Definition 3.17.

a. Suppose $\rho \parallel \sigma \overset{\tau}{\not\longrightarrow}$. If $\rho \dashv \sigma$ then by definition $\rho \xrightarrow{\checkmark}$. Otherwise

$$\rho \dashv \textsc{unfold}(\sigma).$$

Note that $\sigma \overset{\tau}{\not\longrightarrow}$ and therefore by part (i) of Lemma 3.2 it follows that $\textsc{unfold}(\sigma) = \sigma$, which means, since now $\rho \dashv \sigma$, $\rho \xrightarrow{\checkmark}$.

b. Suppose $\rho \parallel \sigma \xrightarrow{\tau} \rho' \parallel \sigma'$. We have to show $(\rho', \sigma') \in \mathcal{R}$, which is obvious if $\rho \dashv \sigma$. On the other hand if $\rho \dashv \textsc{unfold}(\sigma)$ there are three cases, depending on the inference of the action $\rho \parallel \sigma \xrightarrow{\tau} \rho' \parallel \sigma'$. If the action is due to a silent move of ρ, the result follows from part ii of Definition 3.17. In the other cases the result will follow by an application of part i and part ii of Lemma 3.2; and of part ii of Definition 3.17.

□

3.2. *The server pre-order*

In this sub-section we show how to compare servers in terms of their ability to satisfy clients; once more this is done in terms of their respective contracts.

Definition 3.25 (server pre-order). We write $\sigma_1 \sqsubseteq_{\text{SRV}} \sigma_2$ whenever for every contract ρ, $\rho \dashv \sigma_1$ implies $\rho \dashv \sigma_2$.

This provides us with a natural subsumption-like pre-order between server-side contracts. For if $\sigma \sqsubseteq_{\text{SRV}} \sigma'$ then we are assured every client satisfied by a server running the contract σ is by definition also satisfied by a server running the contract σ'.

One consequence of Prepositions 3.23 and 3.24 is that

$$[\![\text{UNFOLD}(\sigma)]\!]_{\text{SRV}} = [\![\sigma]\!]_{\text{SRV}}$$

and therefore when reasoning about the server pre-order we can work up to unfolding.

Notation A. In the rest of the paper we will at times use the symbols $\sum$ and $\bigoplus$ to write contracts, for instance

$$\begin{array}{lll} \sum_{i \in [1;m]} \sigma_i & \text{in place of} & \sigma_1 + \sigma_2 + \cdots + \sigma_m \\ \bigoplus_{i \in [1;n]} \sigma_i & \text{in place of} & \sigma_1 \oplus \sigma_2 \oplus \cdots \oplus \sigma_n \end{array}$$

This is justified by the fact that

$$\begin{array}{ll} \sigma \oplus \rho =_{\text{SRV}} \rho \oplus \sigma & \sigma \oplus (\sigma' \oplus \sigma'') =_{\text{SRV}} (\sigma \oplus \sigma') \oplus \sigma'' \\ \sigma + \rho =_{\text{SRV}} \rho + \sigma & \sigma + (\sigma' + \sigma'') =_{\text{SRV}} (\sigma + \sigma') + \sigma'' \end{array}$$

where $=_{\text{SRV}}$ is the equivalence relation given in the obvious way by $\sqsubseteq_{\text{SRV}}$.

3.2.1. *Co-inductive characterization.*

Here we give a co-inductive characterization of the server pre-order $\sqsubseteq_{\text{SRV}}$; this is based on a number of semantic properties of contracts, which we outline in the following lemmas.

Lemma 3.26. If $\sigma_1 \sqsubseteq_{\text{SRV}} \sigma_2$ and $\sigma_2 \xrightarrow{\tau} \sigma_2'$ then $\sigma_1 \sqsubseteq_{\text{SRV}} \sigma_2'$.

Proof. Suppose $\rho \dashv \sigma_1$, where $\sigma_1 \sqsubseteq_{\text{SRV}} \sigma_2$ and $\sigma_2 \xrightarrow{\tau} \sigma_2'$; we have to show $\rho \dashv \sigma_2'$.

We know $\dashv$ is a co-inductive compliance relation, and also that $\rho \parallel \sigma_2 \xrightarrow{\tau} \rho \parallel \sigma_2'$. So by part (ii) of Definition 3.17 the required $\rho \dashv \sigma_2'$ follows. □

The next property involves the *acceptance sets* of contracts. For any $\text{R} \subseteq Act$, let us write $\sigma \downarrow \text{R}$ whenever $\sigma \overset{\tau}{\not\longrightarrow}$ and $\text{R} = \{\, \alpha \in Act \mid \sigma \xrightarrow{\alpha} \,\}$. These sets $\text{R}, \text{R}', \ldots$ are called *initials* (Eshuis and Fokkinga 2002) or *ready sets* (Laneve and Padovani 2007, 2008). If $\sigma \downarrow \text{R}$ we say that σ *converges to* R; indeed, in our presentation only *stuck* states have ready sets.

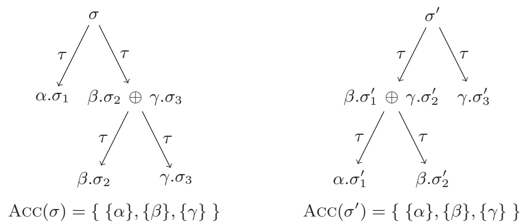

Fig. 8. An example of acceptance sets (see Example 3.28).

Definition 3.27 (acceptance set). For every contract σ let

$$\text{Acc}(\sigma) = \{ \text{R} \mid \sigma \xrightarrow{\tau}{}^{*} \sigma',\ \sigma' \downarrow \text{R} \}.$$

We say that $\text{Acc}(\sigma)$ is the *acceptance sets* of σ.

One sees easily that $\text{NIL} \downarrow \emptyset$ and $\mathbf{1} \downarrow \emptyset$ (as $\checkmark \notin Acts$), so

$$\text{Acc}(\text{NIL}) = \text{Acc}(\mathbf{1}) = \{\emptyset\}.$$

Example 3.28. In Figure 8, we depict the LTS's and the acceptance sets of the ensuing contracts

$$\sigma = \alpha.\sigma_1 \oplus (\beta.\sigma_2 \oplus \gamma.\sigma_3) \quad \sigma' = (\alpha.\sigma'_1 \oplus \beta.\sigma'_2) \oplus \gamma.\sigma'_3.$$

Proposition 3.29. Let $\sigma \in \mathcal{C}$. The ensuing statements are true,

a. if $\sigma \downarrow \text{R}$ then R is finite
b. the set $\text{Acc}(\sigma)$ is finite
c. the set $\text{Acc}(\sigma)$ is non-empty.

Proof. Part a follows from the fact that external sums contain a finite amount of summands. Part b follows form Lemma 3.11. Part c follows from the fact that if $\sigma\downarrow$ then σ has stuck derivatives. □

Example 3.30 (divergent terms and acceptance sets). Here we show the acceptance set of a divergent term. Let $\sigma = \mu x.(\ \alpha.x + x\ )$. For every ready set R we have $\sigma \not\downarrow \text{R}$ because $\sigma \xrightarrow{\tau}$, and the same is true for the derivative $\alpha.\sigma + \sigma$, because it also performs a τ:

$$\sigma \xrightarrow{\tau} \alpha.\sigma + \sigma \xrightarrow{\tau} \ldots$$

We thus conclude that $\text{Acc}(\sigma) = \emptyset$. Indeed, in the proof of part c of Proposition 3.29 the crucial hypothesis is $\sigma\downarrow$.

Individual acceptance sets are compared by their ability to offer interactions. We will write $\text{R} \sqsubseteq \text{S}$ whenever for every $\alpha_{\text{R}} \in \text{R}$ and every β such that $\beta \bowtie \alpha_{\text{R}}$ there exists some action $\alpha_{\text{S}} \in \text{S}$ such that $\beta \bowtie \alpha_{\text{S}}$ also. The precise meaning of this pre-order actually depends on the instantiation of the interaction relation $\bowtie$.

Example 3.31. Suppose that Act be the set $\{ a?, a! \mid a \in A \}$ and $\bowtie_i$ is defined as in Example 3.16. One can check that $\mathrm{R} \sqsubseteq \mathrm{S}$ if and only if $\mathrm{R} \subseteq \mathrm{S}$.

Suppose that

$$Act = \{ ?\mathtt{b}, !\mathtt{b} \mid \mathtt{b} \in \mathtt{BT} \} \cup \{ \mathtt{l}?, \mathtt{l}! \mid \mathtt{l} \in \mathbb{L} \}$$

and $\bowtie_c$ as in Example 3.16. It turns out that $\mathrm{R} \sqsubseteq \mathrm{S}$ whenever the following conditions are true,

— $!\mathtt{l} \in \mathrm{R}$ implies $!\mathtt{l} \in \mathrm{S}$ for every $\mathtt{l} \in \mathrm{L}$.
— $?\mathtt{l} \in \mathrm{R}$ implies $?\mathtt{l} \in \mathrm{S}$ for every $\mathtt{l} \in \mathrm{L}$.
— $?\mathtt{b}_{\mathrm{R}} \in \mathrm{R}$ implies $?\mathtt{b}_{\mathrm{S}} \in \mathrm{S}$ for some type $\mathtt{b}_{\mathrm{S}}$ such that $\mathtt{b}_{\mathrm{S}} \preccurlyeq_{\mathtt{g}} \mathtt{b}_{\mathrm{R}}$.
— $!\mathtt{b}_{\mathrm{R}} \in \mathrm{R}$ implies $!\mathtt{b}_{\mathrm{S}} \in \mathrm{S}$ for some type $\mathtt{b}_{\mathrm{S}}$ such that $\mathtt{b}_{\mathrm{R}} \preccurlyeq_{\mathtt{g}} \mathtt{b}_{\mathrm{S}}$.

Lemma 3.32. Suppose $\sigma_1 \sqsubseteq_{\mathrm{SRV}} \sigma_2$ and $\sigma_2 \Downarrow \mathrm{R}$, then there is some $\mathrm{R}' \in \mathrm{ACC}(\sigma_1)$ such that $\mathrm{R}' \sqsubseteq \mathrm{R}$.

Proof. This proof proceeds by contradiction. To establish the contradiction we construct a contract ρ such that

a. $\rho \dashv \sigma_1$.
b. $\rho \not\dashv \sigma_2$.

Suppose there is no $\mathrm{R}' \in \mathrm{ACC}(\sigma_1)$ such that $\mathrm{R}' \sqsubseteq \mathrm{R}$; thanks to part c of Proposition 3.29 this fact cannot be true because $\mathrm{ACC}(\sigma_1)$ is empty. Again by Proposition 3.29 we know $\mathrm{ACC}(\sigma_1)$ to be finite, so let $\mathrm{R}_1, \ldots, \mathrm{R}_n$ be all the elements in $\mathrm{ACC}(\sigma_1)$. From the hypothesis there are $\alpha_i \in \mathrm{R}_i$ and $\beta_i \bowtie \alpha_i$ such that $\beta_i \not\bowtie \alpha$ whenever $\alpha \in \mathrm{R}'$. Let the contract ρ be defined as $\beta_1.\mathbf{1} + \cdots + \beta_n.\mathbf{1}$.

First notice that b above is true: since $\sigma_2 \Downarrow \mathrm{R}$, $\rho \parallel \sigma_2 \xrightarrow{\tau}{}^{*} \rho \parallel \sigma_2'$ such that $\rho \parallel \sigma_2' \stackrel{\tau}{\not\longrightarrow}$ and $\rho \stackrel{\checkmark}{\not\longrightarrow}$. This means that (ρ, σ_2) cannot be contained in any co-inductive compliance relation.

To establish a above it is sufficient to prove that

$$\mathcal{R} = \{ (\rho, \sigma_1') \mid \sigma_1 \xrightarrow{\tau}{}^{*} \sigma_1' \}$$

is a co-inductive compliance relation, which is relatively straightforward. □

Yet another property of $\sqsubseteq_{\mathrm{SRV}}$ will be necessary to give its co-inductive characterization; to prove this property, one more notion is in order.

Definition 3.33 (after).
For any action $\alpha \in Act$ and contract σ, let $(\sigma \text{ AFTER } \alpha)$ be the set

$$\{ \sigma' \mid \sigma \xrightarrow{\tau}{}^{*} \xrightarrow{\beta} \xrightarrow{\tau}{}^{*} \sigma', \text{where } \alpha \bowtie \beta \}.$$

The proof of the following property is immediate.

Proposition 3.34. Let σ be a contract, then for every $\alpha \in Act$, the set $(\sigma \text{ AFTER } \alpha)$ is finite.

Proof. This follows because for every contract $S(\sigma)$ and $F(\sigma)$ are finite; see Lemmas 3.5 and 3.11. □

Note that in general the set (σ AFTER α) may be empty. When it is non-empty, we use $\bigoplus(\sigma \text{ AFTER } \alpha)$ to denote the internal sum of all its elements.

Lemma 3.35. Suppose $\sigma_1 \sqsubseteq_{\text{SRV}} \sigma_2$ and $\sigma_2 \xrightarrow{\beta} \sigma_2'$. Then whenever $\alpha \bowtie \beta$,

a. the set (σ_1 AFTER α) is non-empty
b. the contract $\bigoplus(\sigma_1 \text{ AFTER } \alpha)$ is smaller than σ_2'. Formally,

$$\bigoplus(\sigma_1 \text{ AFTER } \alpha) \ \sqsubseteq_{\text{SRV}} \ \sigma_2'.$$

Proof. To prove of part a, consider the contract $\rho = \mathbf{1} + \alpha.\text{NIL}$. Since,

$$\rho \parallel \sigma_2 \xrightarrow{\tau} \text{NIL} \parallel \sigma_2'$$

it follows that (ρ, σ_2') cannot be in any co-inductive compliance relation, hence $\rho \not\dashv \sigma_2$. Therefore, from $\sigma_1 \sqsubseteq_{\text{SRV}} \sigma_2$, we have that $\rho \not\dashv \sigma_1$.

But because of the construction of ρ this can only be the case if (σ_1 AFTER α) is non-empty. More specifically, if it was empty we could construct a simple co-inductive compliance relation containing the pair (ρ, σ_1).

Now consider part b. Suppose $\rho \dashv \bigoplus(\sigma \text{ AFTER } \alpha)$; we have to show $\rho \dashv \sigma_2'$. To do so consider the contract $\rho' = \mathbf{1} + \alpha.\rho$. Suppose we could establish

$$\rho' \dashv \sigma_1. \tag{1}$$

Because $\sigma_1 \sqsubseteq_{\text{SRV}} \sigma_2$ this would mean that $\rho' \dashv \sigma_2$, from which the required $\rho \dashv \sigma_2'$ follows, by part (ii) of Definition 3.17.

It remains to prove (1) above. Let

$$\mathcal{R} = \{\, (\rho, \sigma') \mid \rho \dashv \sigma',\ \sigma' \in \mathcal{C} \,\} \ \cup \ \{\, (\rho', \sigma_1') \mid \sigma_1 \xrightarrow{\tau}{}^{*} \sigma_1' \,\}.$$

Then, because $\rho \dashv \bigoplus(\sigma \text{ AFTER } \alpha)$, it is easy to establish that $\mathcal{R}$ is a co-inductive compliance relation. □

We have now assembled all the required properties for our co-inductive characterization of the server pre-order.

Definition 3.36 (semantic sub-server relation). Let $\mathcal{F}_{\preccurlyeq_{\text{SRV}}} : \mathcal{P}(\mathcal{C}^2) \longrightarrow \mathcal{P}(\mathcal{C}^2)$ be the function defined so that $(\sigma_1, \sigma_2) \in \mathcal{F}_{\preccurlyeq_{\text{SRV}}}(\mathcal{R})$ if and only if the following conditions hold:

i. if $\sigma_2 \xrightarrow{\tau} \sigma_2'$ then $(\sigma_1, \sigma_2') \in \mathcal{R}$
ii. for every $\text{R} \in \text{ACC}(\sigma_2)$, $\sigma_2 \downarrow \text{R}$ implies $\text{R}' \sqsubseteq \text{R}$ for some $\text{R}' \in \text{ACC}(\sigma_1)$
iii. if $\sigma_2 \xrightarrow{\beta} \sigma_2'$ and $\alpha \bowtie \beta$ then $(\sigma_1 \text{ AFTER } \alpha) \neq \emptyset$ and $\bigoplus(\sigma_1 \text{ AFTER } \alpha) \ \mathcal{R} \ \sigma_2'$.

If $\mathcal{R} \subseteq \mathcal{F}_{\preccurlyeq_{\text{SRV}}}.(\mathcal{R})$ then we say that $\mathcal{R}$ is a co-inductive *semantic sub-server* relation. Let $\preccurlyeq_{\text{SRV}}$ denote the greatest solution of the equation $X = \mathcal{F}_{\preccurlyeq_{\text{SRV}}}.(X)$; formally,

$$\preccurlyeq_{\text{SRV}} = \nu \mathcal{F}_{\preccurlyeq_{\text{SRV}}}.$$

We call $\preccurlyeq_{\text{SRV}}$ the *semantic sub-server* relation.

Proposition 3.37. If $\sigma_1 \sqsubseteq_{\text{SRV}} \sigma_2$ then $\sigma_1 \preccurlyeq_{\text{SRV}} \sigma_2$.

Proof. It is sufficient to prove that the relation $\sqsubseteq_{\text{SRV}}$ is a semantic sub-server relation; this is straightforward in view of the last three lemmas. □

We also have the converse.

Theorem 3.38 (co-inductive characterization). The server pre-order is the greatest semantic sub-server relation.

Proof. We are required to prove that for all contracts σ_1 and σ_2, the following is true,

$$\sigma_1 \sqsubseteq_{\text{SRV}} \sigma_2 \text{ if and only if } \sigma_1 \preccurlyeq_{\text{SRV}} \sigma_2.$$

Because of the previous proposition, it is sufficient to prove that $\sigma_1 \preccurlyeq_{\text{SRV}} \sigma_2$ and $\rho \dashv \sigma_1$ implies $\rho \dashv \sigma_2$. This will follow if we can show that the relation

$$\mathcal{R} = \{ (\rho, \sigma) \mid \rho \dashv \sigma_1 \text{ for some } \sigma_1 \text{ such that } \sigma_1 \preccurlyeq_{\text{SRV}} \sigma \}$$

is a co-inductive compliance relation.

Suppose $\rho \mathrel{\mathcal{R}} \sigma$. By Definition 3.17, we are required to show two things; namely that

a. if $\rho \parallel \sigma \overset{\tau}{\not\longrightarrow}$ then $\rho \overset{\checkmark}{\longrightarrow}$
b. if $\rho \parallel \sigma \overset{\tau}{\longrightarrow} \rho' \parallel \sigma'$ then $(\rho', \sigma') \in \mathcal{R}$.

By the definition of $\mathcal{R}$, we know that there is some contract σ_1 such that $\sigma_1 \preccurlyeq_{\text{SRV}} \sigma$ and $\rho \dashv \sigma_1$.

We prove the point a. If $\rho \parallel \sigma \overset{\tau}{\not\longrightarrow}$ then $\rho \overset{\tau}{\not\longrightarrow}$, $\sigma \overset{\tau}{\not\longrightarrow}$; in addition, the two contracts cannot interact, that is $\rho \overset{\alpha}{\longrightarrow}$ and $\sigma \overset{\beta}{\longrightarrow}$ implies $\alpha \not\bowtie \beta$. Since ρ and σ are stable both $\text{Acc}(\rho)$ and $\text{Acc}(\sigma)$ contain exactly one set each, say R and S respectively. Then rephrasing the above remark we know

$$\alpha \in \text{R}, \beta \in \text{S} \text{ implies } \alpha \not\bowtie \beta. \tag{2}$$

Since $\sigma_1 \preccurlyeq_{\text{SRV}} \sigma$, by part (ii) of Definition 3.36 $\sigma_1 \overset{\tau}{\longrightarrow}^* \sigma_1'$ for some σ_1' such that $\sigma_1' \downarrow \text{S}'$ and $\text{S}' \sqsubseteq \text{S}$. One can use (2) above to show that this means

$$\alpha \in \text{R}, \beta \in \text{S}' \text{ implies } \alpha \not\bowtie \beta.$$

Also, since $\rho \dashv \sigma_1$ and $\sigma_1 \overset{\tau}{\longrightarrow}^* \sigma_1'$, part ii of Definition 3.17 implies $\rho \dashv \sigma_1'$. But $\rho \parallel \sigma_1' \overset{\tau}{\not\longrightarrow}$ and therefore we have the required $\rho \overset{\checkmark}{\longrightarrow}$.

To prove point b above, we have to show that if $\rho \parallel \sigma \overset{\tau}{\longrightarrow} \rho' \parallel \sigma'$ there exists a $\hat{\sigma}$ such that

$$\rho' \dashv \hat{\sigma}, \quad \hat{\sigma} \preccurlyeq_{\text{SRV}} \sigma'.$$

We proceed by case analysis on the rule used to infer $\rho \parallel \sigma \overset{\tau}{\longrightarrow} \rho' \parallel \sigma'$. There are three possibilities: first suppose the inference rule [p-Sil-L] from Figure 7 is used; the premises of the rule imply that $\rho \overset{\tau}{\longrightarrow} \rho'$, and so $\sigma' = \sigma$. In this case the required $\hat{\sigma}$ is σ_1; the definition of $\mathcal{R}$ gives $\sigma_1 \preccurlyeq_{\text{SRV}} \sigma$ and part ii of Definition 3.17 gives $\rho' \dashv \sigma_1$.

The case when the rule [p-Sil-R] is used is similar; choosing the required $\hat{\sigma}$ to be σ_1 again is justified by point (i) of Definition 3.36.

Finally suppose [P-SYNCH] is employed. Now we know that

$$\rho \xrightarrow{\delta} \rho', \quad \sigma \xrightarrow{\beta} \sigma', \quad \delta \bowtie \beta.$$

In this case, we show that the required $\hat{\sigma}$ is $\bigoplus(\sigma_1 \text{ AFTER } \delta)$. Part (iii) of Definition 3.36 implies that $\bigoplus(\sigma_1 \text{ AFTER } \delta) \preccurlyeq_{\text{SRV}} \sigma'$ and thus it suffices to show that $\rho' \dashv \bigoplus(\sigma_1 \text{ AFTER } \delta)$.

The set $\sigma_1 \text{ AFTER } \delta$ is finite and therefore by Lemma 3.22, it is sufficient to prove $\rho' \dashv \sigma''$ for every $\sigma'' \in (\sigma_1 \text{ AFTER } \delta)$.

For such a σ'' we can derive the transition

$$\rho \parallel \sigma_1 \xrightarrow{\tau}{}^{*} \rho' \parallel \sigma'',$$

where one of the reductions is due to the interaction through δ. From part ii of Definition 3.17 it follows that $\rho \dashv \sigma''$. □

3.3. *Must testing*

The compliance relation between contracts, Definition 3.17, has much in common with the idea of testing from Nicola and Hennessy (1984). Here we explain the relationship. We recall the definition of MUST testing, and explain how it differs from the compliance relation. Despite this difference we then go on to show that the testing pre-order, it induces on contracts coincides with the server pre-order.

For every contract ρ and σ a sequence of reductions

$$\rho \parallel \sigma \xrightarrow{\tau} \rho_1 \parallel \sigma_1 \xrightarrow{\tau} \rho_2 \parallel \sigma_2 \longrightarrow \ldots$$

is called a *computation* of $\rho \parallel \sigma$ and each derivative $\rho_i \parallel \sigma_i$ is a *state* of the computation. Intuitively, viewing ρ as a test we say that the state $\rho_i \parallel \sigma_i$ is *successful* if $\rho_i \xrightarrow{\checkmark}$; and a computation is successful if it contains a successful state.

Example 3.39. For every σ all the computations of

$$\mathbf{1} + ?\mathtt{l}_1.\text{NIL} \parallel \sigma$$

are successful, because in the first state we have $\mathbf{1} + ?\mathtt{l}_1.\text{NIL} \xrightarrow{\checkmark}$. On the other hand, suppose that a contract ρ does not perform ✓ and neither do its derivatives. Then no computation of $\rho \parallel \sigma$ is successful.

A computation is *maximal* if either

i. it is infinite, or
ii. it is finite and the last state is stuck, that is has the form $\rho_k \parallel \sigma_k$ where $\rho_k \parallel \sigma_k \stackrel{\tau}{\not\longrightarrow}$.

Definition 3.40 (must testing).
For all contracts ρ, σ we write $\sigma \text{ MUST } \rho$ if all the maximal computations of $\rho \parallel \sigma$ are successful.

The notion of a client contract complying with a server contract differs in two ways from that of a server contract passing a client contract viewed as a test.

Example 3.41. One difference between MUST and $\dashv$ is what happens after a contract has passed a test, that is the test has reached a successful state; the subsequent computation is disregarded by MUST, whereas the compliance relation has to hold for *all* the states in a computation.

As an example consider $\sigma = ?\mathtt{Real}.\textsc{nil}$ and $\rho = \mathbf{1} + !\mathtt{Int}.\textsc{nil}$. Clearly $\rho \xrightarrow{\checkmark}$, and therefore $\sigma \textsc{ must } \rho$, because each maximal computation of $\rho \parallel \sigma$ begins in a successful state. However, $\rho \not\dashv \sigma$ because $\rho \parallel \sigma \xrightarrow{\tau} \textsc{nil} \parallel \textsc{nil}$.

Example 3.42. The second difference is that the compliance relation does not require the testing contract to ever report success, provided that the communication between the contracts can continue indefinitely. As an example consider the following contracts

$$\sigma = \mu x.\, !\mathtt{Bool}.x, \quad \rho = (\,\mu y.\, ?\mathtt{Rnd}.y\,) + !\mathtt{Int}.\mathbf{1}$$

Plainly, one sees that $\rho \not\xrightarrow{\checkmark}$, and, therefore, $\rho \parallel \sigma$ is not a successful state. The only computation of $\rho \parallel \sigma$ is the infinite loop $\rho \parallel \sigma \xrightarrow{\tau} \rho \parallel \sigma$, and therefore $\rho \dashv \sigma$ holds; on the other hand $\sigma \textsc{ must } \rho$ is false. Example 3.19 contains an even simpler instance of the difference between the relation $\dashv$ and the relation MUST.

The MUST relation can be used to define a well-known pre-order:

Definition 3.43 (must pre-order Nicola and Hennessy (1984)).
Let σ_1, σ_2 be contracts; we write $\sigma_1 \sqsubseteq_{\textsc{must}} \sigma_2$ if and only if for every ρ, $\sigma_1 \textsc{ must } \rho$ implies $\sigma_2 \textsc{ must } \rho$.

Notation B. As we discussed in paragraph Notation A of Section 3.2, we use $\bigoplus$ and $\sum$ also when reasoning on the must pre-order. Formally, this is justified by the ensuing equalities,

$$\begin{array}{ll} \sigma \oplus \rho =_{\textsc{must}} \rho \oplus \sigma & \sigma \oplus (\sigma' \oplus \sigma'') =_{\textsc{must}} (\sigma \oplus \sigma') \oplus \sigma'' \\ \sigma + \rho =_{\textsc{must}} \rho + \sigma & \sigma + (\sigma' + \sigma'') =_{\textsc{must}} (\sigma + \sigma') + \sigma'' \end{array}$$

where $=_{\textsc{must}}$ is the equivalence relation given in the obvious manner by $\sqsubseteq_{\textsc{must}}$.

Notwithstanding the differences between must testing and compliance exposed in Examples 3.41 and 3.42, it turns out that the server pre-order $\sqsubseteq_{\textsc{srv}}$ and the MUST pre-order $\sqsubseteq_{\textsc{must}}$ coincide (Corollary 3.51).

First, in a series of lemmas, we show that $\sqsubseteq_{\textsc{must}}$ satisfies the three defining properties of the semantic sub-server relation (Definition 3.36).

Lemma 3.44. Let $\sigma_1 \sqsubseteq_{\textsc{must}} \sigma_2$. If $\sigma_2 \xrightarrow{\tau} \sigma_2'$ then $\sigma_1 \sqsubseteq_{\textsc{must}} \sigma_2'$.

Proof. Take a contract ρ such that $\sigma_1 \textsc{ must } \rho$ and a maximal computation C performed by $\sigma_2' \parallel \rho$. It is easy to see that a maximal computation from $\sigma_2 \parallel \rho$ can be obtained by prefixing C with the move $\sigma_2 \parallel \rho \xrightarrow{\tau} \sigma_2' \parallel \rho$.

Since $\sigma_1 \sqsubseteq_{\textsc{must}} \sigma_2$,it follows that this extended computation must be successful. However, this implies that C itself is successful since ρ does not change during the initial extending move. □

Lemma 3.45. Let $\sigma_1 \sqsubseteq_{\text{MUST}} \sigma_2$ and $\sigma_2 \Downarrow \text{R}$. There exists a $\text{R}' \in \text{Acc}(\sigma_1)$ such that $\text{R}' \sqsubseteq \text{R}$.

Proof. The proof is similar to that of Lemma 3.32, and proceeds by contradiction. For some $n \geqslant 1$, let

$$\text{Acc}(\sigma_1) = \{ \text{R}_1, \ldots, \text{R}_n \}$$

and suppose that $\text{R}_i \not\sqsubseteq \text{R}$. This means that for every R_i there is $\alpha_i \in \text{R}_i$ and a β_i such that $\beta_i \bowtie \alpha_i$ and $\beta_i \not\bowtie \alpha$ whenever $\alpha \in \text{R}$.

Let ρ be the contract $\beta_1.\mathbf{1} + \cdots + \beta_n.\mathbf{1}$. The contradiction is established by showing that

a. σ_1 MUST ρ while
b. σ_2 MUST ρ is false.

Both of which we leave to the reader. Intuitively b follows because $\sigma_2 \Downarrow \text{R}$, while a is a consequence of the fact that if $\sigma_1 \xrightarrow{\tau}{}^{*} \sigma' \stackrel{\tau}{\not\longrightarrow}$ then $\sigma' \Downarrow \text{R}_i$ for some $1 \leqslant i \leqslant n$. ☐

Lemma 3.46. Let $\sigma_1 \sqsubseteq_{\text{MUST}} \sigma_2$ and $\sigma_2 \xrightarrow{\beta} \sigma_2'$. Whenever $\alpha \bowtie \beta$

a. the set (σ_1 AFTER α) is not empty
b. the contract $\bigoplus(\sigma_1 \text{ AFTER } \alpha)$ is smaller than σ_2'. Formally,

$$\bigoplus(\sigma_1 \text{ AFTER } \alpha) \sqsubseteq_{\text{MUST}} \sigma_2'$$

Proof. The proof of part a is analogous to that of part a in Lemma 3.35, but the contract to be used in this case is $\rho = (\mathbf{1} \oplus \mathbf{1}) + \alpha.\text{NIL}$.

We prove point part b by contradiction. Suppose there is a contract ρ' such that we have $\bigoplus(\sigma_1 \text{ AFTER } \alpha)$ MUST ρ', while σ_2' MUST ρ' is false. Consider the contract $\rho = \alpha.\rho' \oplus \mathbf{1}$. Clearly σ_2 MUST ρ is false while σ_1 MUST ρ is true. This contradicts the hypothesis that $\sigma_1 \sqsubseteq_{\text{MUST}} \sigma_2$. ☐

Proposition 3.47. If $\sigma_1 \sqsubseteq_{\text{MUST}} \sigma_2$ then $\sigma_1 \preccurlyeq_{\text{SRV}} \sigma_2$.

Proof. The previous three lemmas show that $\sqsubseteq_{\text{MUST}}$ is a semantic sub-server relation, from which the result follows. ☐

To establish the converse of this result we need to develop some additional notation. The first is a generalization of the relation σ AFTER α to σ AFTER u where u is a non-empty sequence of actions from $Act^\star$. This is defined by induction on the length of u, with the inductive case being

$$(\sigma \text{ AFTER } w\alpha) = \bigcup_{\sigma' \in (\sigma \text{ AFTER } w)} (\sigma' \text{ AFTER } \alpha).$$

Example 3.48. Let $\sigma = !\mathtt{t}_1.(?\mathtt{t}_2.\sigma_1 + ?\mathtt{t}_3.\sigma_2) + !\mathtt{t}_1.\text{NIL}$ and $?\mathtt{t}_3 \bowtie !\mathtt{t}_2$; we have the following equalities,

$$\begin{aligned}(\sigma \text{AFTER } ?\mathtt{t}_1!\mathtt{t}_2) &= \textstyle\bigcup_{\sigma' \in (\sigma \text{ AFTER } ?\mathtt{t}_1)} (\sigma' \text{ AFTER } !\mathtt{t}_2) \\ &= \textstyle\bigcup_{\sigma' \in \{\text{NIL}, ?\mathtt{t}_2.\sigma_1 + ?\mathtt{t}_3.\sigma_2\}} (\sigma' \text{ AFTER } !\mathtt{t}_2) \\ &= \{\sigma_1, \sigma_2\}.\end{aligned}$$

Next we generalize the interaction relation $\alpha \bowtie \beta$ to non-empty sequences, $u \bowtie w$ in the obvious manner; note that this implies that u and w have the same length. Finally we need the notion of contracts performing sequence of actions. For $u \in Act^\star$ let $\sigma \stackrel{u}{\Longrightarrow} \sigma'$ be the least relation which satisfies

a. $\sigma \stackrel{\varepsilon}{\Longrightarrow} \sigma$ for every contract σ
b. $\sigma \stackrel{u}{\Longrightarrow} \sigma_1$, $\sigma_1 \stackrel{a}{\longrightarrow} \sigma'$, where $a \in Act$, implies $\sigma \stackrel{ua}{\Longrightarrow} \sigma'$
c. $\sigma \stackrel{u}{\Longrightarrow} \sigma_1$, $\sigma_1 \stackrel{\tau}{\longrightarrow} \sigma'$ implies $\sigma \stackrel{u}{\Longrightarrow} \sigma'$.

We have the obvious generalization of condition (iii) in Definition 3.36:

Lemma 3.49. Suppose $\sigma_1 \;\mathcal{R}\; \sigma_2$ for some semantic sub-server relation $\mathcal{R}$, and $\sigma_2 \stackrel{u}{\Longrightarrow} \sigma_2'$ for some non-empty $u \in Act^\star$. Then $v \bowtie u$ implies that

a. the set $(\sigma_1 \text{ AFTER } v)$ is not empty
b. the contract $\bigoplus(\sigma_1 \text{ AFTER } v)$ is related by $\mathcal{R}$ to σ_2'. Formally,

$$\bigoplus(\sigma_1 \text{ AFTER } v) \;\mathcal{R}\; \sigma_2'.$$

Proof. By induction on the non-empty size of u; the base case follows from part (iii) of Definition 3.36. □

Theorem 3.50 (co-inductive characterization). The must pre-order is the greatest semantic sub-server relation.

Proof. We have to prove that for all contracts σ_1, σ_2

$$\sigma_1 \preccurlyeq_{\text{SRV}} \sigma_2 \text{ if and only if } \sigma_1 \sqsubseteq_{\text{MUST}} \sigma_2.$$

Because of Proposition 3.47 it is sufficient to prove $\sigma_1 \preccurlyeq_{\text{SRV}} \sigma_2$ implies $\sigma_1 \sqsubseteq_{\text{MUST}} \sigma_2$. So, suppose $\sigma_1 \preccurlyeq_{\text{SRV}} \sigma_2$ and $\sigma_1 \text{ MUST } \rho$; we must prove $\sigma_2 \text{ MUST } \rho$.

Consider a maximal computation of $\sigma_2 \parallel \rho$

$$\sigma_2 \parallel \rho \stackrel{\tau}{\longrightarrow} \sigma_2^1 \parallel \rho_1 \stackrel{\tau}{\longrightarrow} \ldots \tag{3}$$

We first examine the case when this is finite, with terminal state $\sigma_2^k \parallel \rho_k$. Intuitively this finite computation can be unzipped to give the contributions from the individual components σ_2 and ρ:

$$\sigma_2 \stackrel{u}{\Longrightarrow} \sigma_2^k \qquad \rho \stackrel{v}{\Longrightarrow} \rho_k. \qquad\qquad \text{where } v \bowtie u$$

We are required to show that one of the derivatives of ρ in $\rho \stackrel{v}{\Longrightarrow} \rho_k$ is successful. To this aim we will exhibit a suitable computation of $\sigma_1 \parallel \rho$; in particular we will show that there exists a σ_1' such that

a. the composition $\sigma_1' \parallel \rho_k$ is stuck
b. the computation $\sigma_1 \parallel \rho \stackrel{\tau}{\longrightarrow}^* \sigma_1' \parallel \rho_k$ exists
c. the derivatives of ρ in the computation of point (a) are contained in the computation $\sigma_2 \parallel \rho \stackrel{\tau}{\longrightarrow}^* \sigma_2^k \parallel \rho_k$.

These three points are enough to prove that in $\rho \stackrel{v}{\Longrightarrow}$ there exists a successful derivative: thanks to (a), the computation in (b) is a maximal computation of $\sigma_1 \parallel \rho$; the assumption that σ_1 MUST ρ implies that the computation in (b) contains a successful $\hat{\rho}$, and point (c) ensures that $\hat{\rho}$ is contained in $\rho \stackrel{v}{\Longrightarrow}$.

We prove one by one the points above.

a. Here we show that, for a suitable σ_1', the composition $\sigma_1' \parallel \rho_k$ is stuck.
By assumption the state $\sigma_2^k \parallel \rho_k$ is terminal; this implies that (1) both σ_2^k and ρ_k are stuck. A consequence is that their acceptance sets are singleton; say $\mathrm{A{\scriptsize CC}}(\sigma_2^k) = \{\mathrm{R}\}$ and $\mathrm{A{\scriptsize CC}}(\rho_k) = \{\mathrm{S}\}$; and (2) the contracts σ_2^k and ρ_k cannot interact. Formally

$$\alpha \in \mathrm{R} \text{ implies } \beta \not\bowtie \alpha \text{ for every } \beta \in \mathrm{S}.$$

Consider now the contract $\bigoplus(\sigma_1 \text{ AFTER } v)$; part a of Lemma 3.49 implies that the set $(\sigma_1 \text{ AFTER } v)$ is not empty and part b of the same lemma implies that $\bigoplus(\sigma_1 \text{ AFTER } v) \preccurlyeq_{\mathrm{SRV}} \sigma_2^k$.
Part (ii) of Definition 3.36 and (2) above imply that there exists a set

$$\mathrm{R}' \in \mathrm{A{\scriptsize CC}}(\bigoplus(\sigma_1 \text{ AFTER } v))$$

such that

$$\alpha \in \mathrm{R}' \text{ implies } \beta \not\bowtie \alpha \text{ for every } \beta \in \mathrm{S}'. \tag{4}$$

From Definition 3.27, it follows that there exists a contract σ_1' such that $\sigma_1' \downarrow \mathrm{R}'$ and

$$\bigoplus(\sigma_1 \text{ AFTER } v) \stackrel{\tau}{\longrightarrow}{}^{*} \sigma_1'.$$

The latter fact means that $\sigma_1' \stackrel{\tau}{\not\longrightarrow}$ and (4) above means that σ_1' and ρ_k cannot interact; Since (1) above proves that ρ_k is stuck we have shown that $\sigma_1' \parallel \rho_k$ is stuck.

b. We are required to exhibit the computation $\sigma_1 \parallel \rho \stackrel{\tau}{\longrightarrow}{}^{*} \sigma_1' \parallel \rho_k$.
Since $\bigoplus(\sigma_1 \text{ AFTER } v) \stackrel{\tau}{\longrightarrow}{}^{*} \sigma_1'$, there exists a $\sigma_1'' \in (\sigma_1 \text{ AFTER } v)$ such that $\sigma_1'' \stackrel{\tau}{\longrightarrow}{}^{*} \sigma_1'$. From the definition of $(\sigma_1 \text{ AFTER } v)$, it follows that $\sigma_1 \stackrel{w}{\Longrightarrow} \sigma_1''$ for some $w \in Act^{\star}$ such that $w \bowtie v$, and this implies that $\sigma_1 \stackrel{w}{\Longrightarrow} \sigma_1'$. Zipping this action sequence together with $\rho \stackrel{v}{\Longrightarrow} \rho_k$ we obtain the computation

$$\sigma_1 \parallel \rho \stackrel{\tau}{\longrightarrow}{}^{*} \sigma_1' \parallel \rho_k \stackrel{\tau}{\not\longrightarrow}.$$

We remark that the computation above is finite and cannot be extended, hence it is maximal.

c. The derivatives of ρ in the computation

$$\sigma_1 \parallel \rho \stackrel{\tau}{\longrightarrow}{}^{*} \sigma_1' \parallel \rho_k$$

are contained in the computation $\sigma_2 \parallel \rho \stackrel{\tau}{\longrightarrow}{}^{*} \sigma_2^k \parallel \rho_k$ because the former computation has been obtained by zipping $\rho \stackrel{v}{\Longrightarrow} \rho_k$ with a computation made by σ_1.

Now suppose that the maximal computation (3) above is infinite. Then the result of unzipping gives infinite traces u, v such that

$$\sigma_2 \stackrel{u}{\Longrightarrow} \quad \rho \stackrel{v}{\Longrightarrow}.$$

Let us denote the finite prefixes of these traces of length k by $u(k)$, $v(k)$ respectively. By Lemma 3.49, we know that σ_1 AFTER $v(k)$ is non-empty, for every $k \geqslant 0$. This means that the LTS generated by σ_1 is infinite.

Now consider the sub-LTS consisting of all nodes σ which can be reached from σ_1 using a weak move $\sigma_1 \stackrel{w(k)}{\Longrightarrow} \sigma'$, where $w(k)$ is some trace satisfying $w(k) \bowtie v(k)$. This sub-LTS is therefore infinite. It is also finite-branching and so by Kőnig's lemma it has an infinite path. By following this path from the root we get $\sigma_1 \stackrel{w}{\Longrightarrow}$ such that $w(k) \bowtie v(k)$, for every $k \geqslant 0$.

This infinite computation can now be zipped with $\rho \stackrel{v}{\Longrightarrow}$ to obtain an infinite computation from $\sigma_1 \parallel \rho$. Since σ_1 MUST ρ it follows that there is some $\sigma_2^k \parallel \rho_k$ in the maximal computation (3) above which is successful, and therefore σ_2 MUST ρ. □

Corollary 3.51. The server pre-order equals the must pre-order. Formally

$$\sqsubseteq_{\text{SRV}} = \sqsubseteq_{\text{MUST}} .$$

Proof. It is a consequence of Theorems 3.38 and 3.50. □

Thus far, we have never written explicitly the parameter $\bowtie$ used to infer the reductions of parallel compositions of contracts. Now we make the parameter $\bowtie$ explicit; this let us emphasize the true import of Corollary 3.51. For every $\bowtie \subseteq Act^2$, we denote with $\stackrel{\tau}{\longrightarrow}_{\bowtie}$ the relation inferred by using the rule in Figure 7 and the $\bowtie$ in the subscript. By replacing $\stackrel{\tau}{\longrightarrow}_{\bowtie}$ in Definition 3.17 and 3.40, we define two relations $\dashv_{\bowtie}$ and $\text{MUST}_{\bowtie}$; in turns this lets us generalize the pre-orders we have studied,

— for every $\bowtie \subseteq Act^2$, we write $\sigma_1 \sqsubseteq_{\text{SRV}}^{\bowtie} \sigma_2$ if and only if $\rho \dashv^{\bowtie} \sigma_1$ implies that $\rho \dashv^{\bowtie} \sigma_2$;
— for every $\bowtie \subseteq Act^2$, we write $\sigma_1 \sqsubseteq_{\text{MUST}}^{\bowtie} \sigma_2$ if and only if σ_1 $\text{MUST}^{\bowtie}$ ρ implies that σ_2 $\text{MUST}^{\bowtie}$ ρ.

Since in proving Corollary 3.51 we have used no hypothesis on $\bowtie$, we can rephrase it as follows,

$$\text{for every } \bowtie \subseteq Act^2.\ \sqsubseteq_{\text{SRV}}^{\bowtie} = \sqsubseteq_{\text{MUST}}^{\bowtie} .$$

This means that the equality between the server pre-order and the must pre-order does not depend on the $\bowtie$ at hand. That is, the equality holds regardless of the co-action relation used in rule [P-SYNCH].

4. Session contracts

Here we specialize the contract language, to a sub-language which will be the target of our interpretation of the session types from Section 2. This is the topic of the first sub-section. We then go on to re-examine the server pre-order as it applies to this sub-language; in particular we show that it can also be characterized co-inductively, this time using purely syntactical criteria. In the final section, we give a similar co-inductive characterization to a related sub-client pre-order.

$\sigma, \rho \quad ::=$		**Session Contracts**
	$\mathbf{1}$	*success*
	$\sum_{i\in I} ?\mathtt{l}_i.\sigma$	*external choice, I finite, non-empty*
	$\bigoplus_{i\in I} !\mathtt{l}_i.\sigma$	*internal choice, I finite, non-empty*
	$?\mathtt{t}.\sigma$	*input*
	$!\mathtt{t}.\sigma$	*output*
	x	*contract variable*
	$\mu x.\,\sigma$	*recursion*

We impose the additional proviso that in a term the $\mathtt{l}_i$'s are pair-wise different.

Fig. 9. Session contract grammar.

4.1. *Session contracts*

The syntax for the language $L_{\mathcal{SC}}$ is given in Figure 9.

Definition 4.1 (session contracts). We use $\mathcal{SC}$ to denote the set of closed terms σ of $L_{\mathcal{SC}}$ such that if for some $s \in Act^{\star}$, $\sigma \xrightarrow{s}{}^{*} \sigma'$, then $\sigma'\downarrow$. We refer to these terms as *session contracts*.

Note that $\mathcal{SC}$ is a subset of the more general language of contracts $\mathcal{C}$, but

— external choices are restricted to inputs on labels
— internal choices are restricted to outputs on labels.

Note also that NIL is not a session contract. Instead we have chosen $\mathbf{1}$ to be the base contract, for reasons which will become apparent. Moreover, we already reasoned that a server contract $\mathbf{1}$ has the same behaviour as NIL (Proposition 3.21).

Session contracts, due to their restrictive syntax, enjoy some properties which we will use in the next sub-sections, and which we prove now.

Lemma 4.2. Let σ be a session contract. Then

i. $\sigma \xrightarrow{\checkmark}$ if and only if $\sigma = \mathbf{1}$.
ii. $\sigma \xrightarrow{\tau}{}^{*} \xrightarrow{\checkmark}$ if and only if $\mathrm{UNFOLD}(\sigma) = \mathbf{1}$.
iii. $\sigma \xrightarrow{\alpha}$ if and only if $\sigma \neq \mathbf{1}$.

Proof. Part (i) follows from the restrictive syntax of session contract. The proof of part (ii) requires two arguments. The *if* side, $\mathrm{UNFOLD}(\sigma) = \mathbf{1}$ implies $\sigma \xrightarrow{\tau}{}^{*} \xrightarrow{\checkmark}$, is justified by part (ii) of Lemma 3.2. The *only if* side, $\sigma \xrightarrow{\tau}{}^{*} \xrightarrow{\checkmark}$ implies $\mathrm{UNFOLD}(\sigma) = \mathbf{1}$, can be proven by induction on the length of the sequence $\xrightarrow{\tau}{}^{*}$; the base case being part (i) of this lemma. Part (iii) can be proven by structural induction. □

4.2. *The restricted server pre-order*

Definition 3.25 applies equally well to session contracts, but it is inappropriate as it compares session contracts from the point of view of satisfying clients who may use the more general contracts from Section 3. Instead let us restrict attention to clients who also only run the more restricted session contracts.

Definition 4.3 (restricted server pre-order).
For $\sigma_1, \sigma_2 \in \mathcal{SC}$ let $\sigma_1 \sqsubseteq_{\text{SRV}}^{\mathcal{SC}} \sigma_2$ whenever $\rho \dashv \sigma_1$ implies $\rho \dashv \sigma_2$ for every ρ in $\mathcal{SC}$.

This relation is more generous than $\sqsubseteq_{\text{SRV}}$, in that it allows implementation refinement (Padovani 2010) to happen, as the following example shows.

Example 4.4.

$$?\mathtt{latte}.\mathbf{1} \sqsubseteq_{\text{SRV}}^{\mathcal{SC}} ?\mathtt{moka}.\mathbf{1} + ?\mathtt{latte}.\mathbf{1}$$

If a session contract ρ can interact with $?\mathtt{latte}.\mathbf{1}$ then, modulo unfolding, it has to be defined by an internal sum. Moreover this sum can only contain one summand, and therefore ρ complies also with $?\mathtt{moka}.\mathbf{1} + ?\mathtt{latte}.\mathbf{1}$.

Consider now the more general contract $\rho' = !\mathtt{latte}.\mathbf{1} + !\mathtt{moka}.\text{NIL}$. One can check that

$$\rho' \dashv ?\mathtt{latte}.\mathbf{1}$$

whereas $\rho' \not\dashv ?\mathtt{moka}.\mathbf{1} + ?\mathtt{latte}.\mathbf{1}$. It therefore follows that

$$?\mathtt{latte}.\mathbf{1} \not\sqsubseteq_{\text{SRV}} ?\mathtt{moka}.\mathbf{1} + ?\mathtt{latte}.\mathbf{1}$$

Example 4.5 (e-vote, ballot refinement).
We give a more concrete instance of the previous example. Recall Example 3.14 and consider the session contract

$$\begin{aligned}\text{BallotB} = \mu x.\ ?\mathtt{Login}.(\ !\mathtt{Wrong}.\mathbf{1} \oplus \\ !\mathtt{Ok}.(\ ?\mathtt{VoteA}.\mathbf{1} + ?\mathtt{VoteB}.\mathbf{1} + ?\mathtt{VoteC}.\mathbf{1} + ?\mathtt{VoteD}.\mathbf{1}\ )\ )\end{aligned}$$

BallotB offers to a voter more options than Ballot, and intuitively it should be possible to use a server that guarantees BallotB in place of a server that guarantees Ballot. This is not the case if the contracts are compared with $\sqsubseteq_{\text{SRV}}$, because Ballot $\not\sqsubseteq_{\text{SRV}}$ BallotB. On the other hand, if we restrict our attention to session contracts, and thus to the pre-order $\sqsubseteq_{\text{SRV}}^{\mathcal{SC}}$, we have Ballot $\sqsubseteq_{\text{SRV}}^{\mathcal{SC}}$ BallotB.

When comparing session contracts relative to this pre-order it will be convenient to work modulo unfolding, which is possible because of the following result:

Proposition 4.6. Let σ_1, σ_2 be session contracts; $\sigma_1 \sqsubseteq_{\text{SRV}}^{\mathcal{SC}} \sigma_2$ if and only if $\text{UNFOLD}(\sigma_2) \sqsubseteq_{\text{SRV}}^{\mathcal{SC}} \text{UNFOLD}(\sigma_2)$.

Proof. Follows from Propositions 3.23 and 3.24. □

Proposition 4.7 (bottom element).
The pre-order $\sqsubseteq_{\text{SRV}}^{\mathcal{SC}}$ enjoys the following properties,

i. it has a bottom element
ii. if $\sigma_\perp$ is a bottom element of $\sqsubseteq_{\text{SRV}}^{\mathcal{SC}}$ then $\text{UNFOLD}(\sigma_\perp) = \mathbf{1}$.

Proof. To prove part i we show that $\mathbf{1}$ is a bottom element of $\sqsubseteq_{\text{SRV}}^{\mathcal{SC}}$, that is $\mathbf{1} \sqsubseteq_{\text{SRV}}^{\mathcal{SC}} \sigma$ for every session contract σ. Let ρ be a session contract such that $\rho \dashv \mathbf{1}$. The session contract $\mathbf{1}$ offers no interaction. Therefore, because of the restricted syntax of session contracts,

ρ must also be, modulo unfolding, the simple contract $\mathbf{1}$. Now fix a session contract σ. Clearly $\mathbf{1} \dashv \sigma$, therefore from an application of Proposition 3.24 it follows that $\rho \dashv \sigma$.

To prove part ii let $\sigma_\perp$ be an arbitrary bottom element of $\sqsubseteq^{\mathcal{SC}}_{\text{SRV}}$. We are required to show that $\text{UNFOLD}(\sigma_\perp) = \mathbf{1}$. From the definition of bottom element follows $\sigma_\perp \sqsubseteq^{\mathcal{SC}}_{\text{SRV}} \mathbf{1}$. An application of the previous proposition gives $\text{UNFOLD}(\sigma_\perp) \sqsubseteq^{\mathcal{SC}}_{\text{SRV}} \mathbf{1}$. But now an analysis of the possible syntactic structure of $\text{UNFOLD}(\sigma_\perp)$ quickly yields that it must be $\mathbf{1}$ itself. □

Part ii is relevant because $\mathbf{1}$ is not the only bottom element; for example it is also true that $\mu x.\mathbf{1} \sqsubseteq^{\mathcal{SC}}_{\text{SRV}} \sigma$ for every σ.

We now proceed, as in Section 3.2.1, to give a co-inductive characterization of the restricted server pre-order, this time taking advantage of their restricted syntactic structure.

Definition 4.8 (syntactic sub-server relation).
Let $\mathcal{F}_{\preccurlyeq^{syn}_{\text{SRV}}} : \mathcal{P}(\mathcal{SC}^2) \longrightarrow \mathcal{P}(\mathcal{SC}^2)$ be defined by letting $(\sigma_1, \sigma_2) \in \mathcal{F}_{\preccurlyeq^{syn}_{\text{SRV}}}(\mathcal{R})$ whenever one of the following holds:

i. $\text{UNFOLD}(\sigma_1) = \mathbf{1}$.
ii. $\text{UNFOLD}(\sigma_2) = ?\mathtt{t}_2.\sigma'_2$ and $\text{UNFOLD}(\sigma_1) = ?\mathtt{t}_1.\sigma'_1$ with $\mathtt{t}_1 \preccurlyeq_{\mathsf{g}} \mathtt{t}_2$ and $\sigma'_1 \mathrel{\mathcal{R}} \sigma'_2$.
iii. $\text{UNFOLD}(\sigma_2) = !\mathtt{t}_2.\sigma'_2$ and $\text{UNFOLD}(\sigma_1) = !\mathtt{t}_1.\sigma'_1$ with $\mathtt{t}_2 \preccurlyeq_{\mathsf{g}} \mathtt{t}_1$ and $\sigma'_1 \mathrel{\mathcal{R}} \sigma'_2$.
iv. $\text{UNFOLD}(\sigma_2) = \sum_{j \in J} ?\mathtt{l}_j.\sigma^2_j$ and $\text{UNFOLD}(\sigma_1) = \sum_{i \in I} ?\mathtt{l}_i.\sigma^1_i$ with $I \subseteq J$ and $\sigma^1_i \mathrel{\mathcal{R}} \sigma^2_{i.}$
v. $\text{UNFOLD}(\sigma_2) = \bigoplus_{j \in J} !\mathtt{l}_j.\sigma^2_j$ and $\text{UNFOLD}(\sigma_1) = \bigoplus_{i \in I} !\mathtt{l}_i.\sigma^1_i$ with $J \subseteq I$ and $\sigma^1_j \mathrel{\mathcal{R}} \sigma^2_j$.

If $\mathcal{R} \subseteq \mathcal{F}_{\preccurlyeq^{syn}_{\text{SRV}}}(\mathcal{R})$ then we say that $\mathcal{R}$ is a co-inductive *syntactic sub-server* relation. Let $\preccurlyeq^{syn}_{\text{SRV}}$ denote the greatest solution of the equation $X = \mathcal{F}_{\preccurlyeq^{syn}_{\text{SRV}}}(X)$; formally,

$$\preccurlyeq^{syn}_{\text{SRV}} \; = \nu \mathcal{F}_{\preccurlyeq^{syn}_{\text{SRV}}}$$

We call $\preccurlyeq^{syn}_{\text{SRV}}$ the *syntactic sub-server* relation.

We first show that the set based relation $\sqsubseteq^{\mathcal{SC}}_{\text{SRV}}$ is contained in $\preccurlyeq^{syn}_{\text{SRV}}$; this will follow if we can show that the former satisfies the properties defining the latter.

Lemma 4.9. Let $\sigma_1, \sigma_2 \in \mathcal{SC}$, $\sigma_1 = \text{UNFOLD}(\sigma_1)$, $\sigma_2 = \text{UNFOLD}(\sigma_2)$ and $\sigma_1 \sqsubseteq^{\mathcal{SC}}_{\text{SRV}} \sigma_2$. One of the following is true,

a. if $\sigma_1 = !\mathtt{t}_1.\sigma'_1$ then $\sigma_2 = !\mathtt{t}_2.\sigma'_2$, $\mathtt{t}_2 \preccurlyeq_{\mathsf{g}} \mathtt{t}_1$ and $\sigma'_1 \sqsubseteq^{\mathcal{SC}}_{\text{SRV}} \sigma'_2$
b. if $\sigma_1 = ?\mathtt{t}_1.\sigma'_1$ then $\sigma_2 = ?\mathtt{t}_2.\sigma'_2$, $\mathtt{t}_1 \preccurlyeq_{\mathsf{g}} \mathtt{t}_2$ and $\sigma'_1 \sqsubseteq^{\mathcal{SC}}_{\text{SRV}} \sigma'_2$
c. if $\sigma_1 = \sum_{i \in I} ?\mathtt{l}_\mathtt{i}.\sigma^1_i$ then $\sigma_2 = \sum_{j \in J} ?\mathtt{l}_\mathtt{j}.\sigma^2_j$, $I \subseteq J$ and $\sigma^1_i \sqsubseteq^{\mathcal{SC}}_{\text{SRV}} \sigma^2_i$
d. if $\sigma_1 = \bigoplus_{i \in I} !\mathtt{l}_\mathtt{i}.\sigma^1_i$ then $\sigma_2 = \bigoplus_{j \in J} !\mathtt{l}_\mathtt{j}.\sigma^2_j$ with $J \subseteq I$ and $\sigma^1_j \sqsubseteq^{\mathcal{SC}}_{\text{SRV}} \sigma^2_j$.

Proof. The proof is by case analysis on the structure of σ_1, and the argument depends greatly on the restricted syntax of session contracts. We prove part a and leave the proof of the other parts to the reader.

Suppose $\sigma_1 = !\mathtt{t}_1.\sigma'_1$. Then $?\mathtt{t}_1.\mathbf{1} \dashv \sigma_1$ and because $\sigma_1 \sqsubseteq^{\mathcal{SC}}_{\text{SRV}} \sigma_2$ it follows that $?\mathtt{t}_1.\mathbf{1} \dashv \sigma_2$. Since $?\mathtt{t}_1.\mathbf{1}$ is stuck, σ_2 has to engage in an action $!\mathtt{t}_2$ such that $?\mathtt{t}_1 \bowtie_c !\mathtt{t}_2$; this lets us prove that $\mathtt{t}_2 \preccurlyeq_{\mathsf{g}} \mathtt{t}_1$. In reason of the syntax and the hypothesis $\sigma_2 = \text{UNFOLD}(\sigma_2)$, the equality $\sigma_2 = !\mathtt{t}_2.\sigma'_2$ must hold.

We also have to prove that $\sigma_1' \sqsubseteq^{\mathcal{SC}}_{\text{SRV}} \sigma_2'$. Pick a session contract ρ such that $\rho \dashv \sigma_1'$. Clearly $?\mathtt{t}_1.\rho \dashv \sigma_1$, and thus $?\mathtt{t}_1.\rho \dashv \sigma_2$. Since $?\mathtt{t}_1 \bowtie_c !\mathtt{t}_2$, we apply rule [P-SYNCH] to infer $?\mathtt{t}_1.\rho \parallel \sigma_2 \xrightarrow{\tau} \rho \parallel \sigma_2'$. From the definition of compliance it follows that $\rho \dashv \sigma_2'$. □

Proposition 4.10. For session contracts, $\sigma_1 \sqsubseteq^{\mathcal{SC}}_{\text{SRV}} \sigma_2$ implies $\sigma_1 \preccurlyeq^{syn}_{\text{SRV}} \sigma_2$.

Proof. We prove that $\sqsubseteq^{\mathcal{SC}}_{\text{SRV}}$ is a pre-fixed point of the function $\mathcal{F}_{\preccurlyeq^{syn}_{\text{SRV}}}$ of Definition 4.8, that is $\sigma_1 \sqsubseteq^{\mathcal{SC}}_{\text{SRV}} \sigma_2$ implies $(\sigma_1, \sigma_2) \in \mathcal{F}_{\preccurlyeq^{syn}_{\text{SRV}}}(\sqsubseteq^{\mathcal{SC}}_{\text{SRV}})$.

Suppose $\sigma_1 \sqsubseteq^{\mathcal{SC}}_{\text{SRV}} \sigma_2$. Then by Proposition 4.6 it follows that $\textsc{unfold}(\sigma_1) \sqsubseteq^{\mathcal{SC}}_{\text{SRV}} \textsc{unfold}(\sigma_2)$. Now, if $\textsc{unfold}(\sigma_1) = \mathbf{1}$ by definition $(\sigma_1, \sigma_2) \in \mathcal{F}_{\preccurlyeq^{syn}_{\text{SRV}}}(\sqsubseteq^{\mathcal{SC}}_{\text{SRV}})$. Otherwise we can apply Lemma 4.9 to the pair $\textsc{unfold}(\sigma_1)$, $\textsc{unfold}(\sigma_2)$. This provides the required information to satisfy the requirements ii to v in Definition 4.8, thereby ensuring that $(\sigma_1, \sigma_2) \in \mathcal{F}_{\preccurlyeq^{syn}_{\text{SRV}}}(\sqsubseteq^{\mathcal{SC}}_{\text{SRV}})$. □

Lemma 4.11. Let $\mathcal{R}$ be a co-inductive syntactic sub-server relation and let $\sigma_1 \mathrel{\mathcal{R}} \sigma_2$. If $\sigma_2 \xrightarrow{\tau} \sigma_2'$ then $\sigma_1 \mathrel{\mathcal{R}} \sigma_2'$.

Proof. First note that from Definition 4.8 it follows that

$$\textsc{unfold}(\sigma_1) \mathrel{\mathcal{R}} \textsc{unfold}(\sigma_2). \tag{5}$$

There are two different cases to be discussed, depending on the unfolding of σ_2 being σ_2 itself or not.

a. If $\textsc{unfold}(\sigma_2) \neq \sigma_2$ then $\textsc{unfold}(\sigma_2) = \textsc{unfold}(\sigma_2')$. The equality and (5) above imply $\textsc{unfold}(\sigma_1) \mathrel{\mathcal{R}} \textsc{unfold}(\sigma_2')$; the latter fact means that $\sigma_1 \mathrel{\mathcal{R}} \sigma_2'$.
b. If $\textsc{unfold}(\sigma_2) = \sigma_2$ then σ_2 must be an internal sum, say $\sigma_2 = \bigoplus_{i \in I} !\mathtt{l}_\mathtt{i}.\sigma^2_i$, because the contract σ_2 can perform a silent move and cannot unfold. This implies that σ_2' is the internal sum $\bigoplus_{k \in K} !\mathtt{l}_\mathtt{k}.\sigma^2_k$ for some $K \subseteq I$. From Definition 4.8 it follows that

$$\textsc{unfold}(\sigma_1) = \bigoplus_{j \in J} !\mathtt{l}_\mathtt{j}.\sigma^2_j$$

with $I \subseteq J$. Since, $\textsc{unfold}(\sigma_2') = \sigma_2'$ and $K \subseteq I \subseteq J$ one sees easily that

$$\textsc{unfold}(\sigma_1) \mathrel{\mathcal{R}} \textsc{unfold}(\sigma_2')$$

thus $\sigma_1 \mathrel{\mathcal{R}} \sigma_2$.

□

Lemma 4.12. Let $\mathcal{R}$ be a co-inductive syntactic sub-server relation, $\sigma_1 \mathrel{\mathcal{R}} \sigma_2$ and $\sigma_2 \downarrow \textsc{r}$. There exists a $\textsc{r}' \in \textsc{Acc}(\sigma_1)$ such that $\textsc{r}' \sqsubseteq \textsc{r}$.

Proof. Using Lemma 3.2 we know $\textsc{unfold}(\sigma_2) = \sigma_2$, since $\sigma_2 \not\xrightarrow{\tau}$. From Definition 4.8 it follows that $\textsc{unfold}(\sigma_1) \mathrel{\mathcal{R}} \sigma_2$. Now, according to the cases in Definition 4.8 and a case analysis on the form of σ_2, one can show that $\textsc{unfold}(\sigma_1) \xrightarrow{\tau}{}^* \sigma_1'$ for some σ_1' which satisfies the required properties. We leave the details of the case analysis to the reader.

Finally, the proof that $\sigma_1 \xrightarrow{\tau}{}^{*} \sigma_1'$ amounts in two steps. We apply Lemma 3.2, which ensures that $\sigma_1 \xrightarrow{\tau}{}^{*} \text{UNFOLD}(\sigma_1)$. Now we know that

$$\sigma_1 \xrightarrow{\tau}{}^{*} \text{UNFOLD}(\sigma_1) \xrightarrow{\tau}{}^{*} \sigma_1'$$

so the transitivity of $\xrightarrow{\tau}{}^{*}$ gives the result.

□

Lemmas 4.11 and 4.12 proves that the pre-order $\preccurlyeq^{syn}_{\text{SRV}}$ enjoys two of the properties of the pre-order $\preccurlyeq_{\text{SRV}}$. The third property of $\preccurlyeq_{\text{SRV}}$, though, is not afforded by $\preccurlyeq^{syn}_{\text{SRV}}$.

Example 4.13. Let $\sigma_1 = ?\mathtt{latte}.\mathbf{1}$ and $\sigma_2 = ?\mathtt{latte}.\mathbf{1} + ?\mathtt{moka}.\mathbf{1}$; in Example 4.4 we have shown that $\sigma_1 \preccurlyeq^{syn}_{\text{SRV}} \sigma_2$. On the one hand, we can prove the following things, $\sigma_2 \xrightarrow{?\mathtt{moka}} \mathbf{1}$ and $!\mathtt{moka} \bowtie_c ?\mathtt{moka}$; on the other hand

$$(\,\sigma_1 \text{ AFTER } !\mathtt{moka}\,) = \emptyset.$$

As in Example 4.4, the crucial fact is that the pre-order $\preccurlyeq^{syn}_{\text{SRV}}$ allows implementation refinement (Padovani 2010).

Theorem 4.14 (co-inductive characterization). For session contracts σ_1, σ_2, $\sigma_1 \sqsubseteq^{\mathcal{SC}}_{\text{SRV}} \sigma_2$ if and only if $\sigma_1 \preccurlyeq^{syn}_{\text{SRV}} \sigma_2$.

Proof. The only if part of the theorem is Proposition 4.10 while the if part, that is the set inclusion $\preccurlyeq^{syn}_{\text{SRV}} \subseteq \sqsubseteq^{\mathcal{SC}}_{\text{SRV}}$, follows from the fact that the relation

$$\mathcal{R} = \{\, (\rho, \sigma_2) \mid \sigma_1 \preccurlyeq^{syn}_{\text{SRV}} \sigma_2\,, \rho \dashv \sigma_1 \text{ for some } \sigma_1 \in \mathcal{SC}\,\}$$

contains (ρ, σ_2) and is a compliance. We prove the latter.

We have to show that $\mathcal{R}$ satisfies the two properties in Definition 3.17. Let $(\rho, \sigma) \in \mathcal{R}$; by definition there exists a σ_1 such that $\rho \dashv \sigma_1$ and $\sigma_1 \preccurlyeq^{syn}_{\text{SRV}} \sigma$.

To prove point (i) of Definition 3.17 assume $\rho \parallel \sigma \stackrel{\tau}{\not\longrightarrow}$. This implies that σ and ρ are both stuck, so $\text{Acc}(\rho) = \{\text{S}\}$ and $\text{Acc}(\sigma) = \{\text{R}\}$, and that

$$\alpha \in \text{R} \text{ implies } \beta \not\bowtie_c \alpha \text{ whenever } \beta \in \text{S}.$$

An application of Lemma 4.12 and of the definition of acceptance set gives a σ_1' such that $\sigma_1 \xrightarrow{\tau}{}^{*} \sigma_1' \downarrow \text{R}'$ and $\text{R}' \sqsubseteq \text{R}$. The last inequality implies that

$$\alpha \in \text{R}' \text{ implies } \beta \not\bowtie_c \alpha \text{ whenever } \beta \in \text{S}$$

therefore, $\rho \parallel \sigma_1' \stackrel{\tau}{\not\longrightarrow}$. Part (ii) of Definition 3.17 and the assumption $\rho \dashv \sigma_1$ imply that the session contract ρ complies with σ_1' so $\rho \xrightarrow{\checkmark}$.

What we have left to do now is to show that if $\rho \parallel \sigma \xrightarrow{\tau} \rho' \parallel \sigma'$ then $(\rho', \sigma') \in \mathcal{R}$, that is there exists a $\hat{\sigma} \in \mathcal{SC}$ such that

$$\rho' \dashv \hat{\sigma}, \quad \hat{\sigma} \preccurlyeq^{syn}_{\text{SRV}} \sigma'.$$

Assume $\rho \parallel \sigma \xrightarrow{\tau} \rho' \parallel \sigma'$. The argument depends on the rule used to infer this silent move (see Figure 7). If rule [P-SIL-L] was used then $\sigma' = \sigma$ and $\rho \xrightarrow{\tau} \rho'$; let $\hat{\sigma} = \sigma_1$.

Then we already know that $\hat{\sigma} \preccurlyeq^{syn}_{\text{SRV}} \sigma'$, and part ii of Definition 3.17 implies $\rho' \dashv \hat{\sigma}$. If rule [P-SIL-R] was applied then $\rho' = \rho$ and $\sigma \xrightarrow{\tau} \sigma'$. In this case an application of Lemma 4.11 implies $\sigma_1 \preccurlyeq^{syn}_{\text{SRV}} \sigma'$. We know by assumption that $\rho \dashv \sigma_1$, so the $\hat{\sigma}$ we are looking for is σ_1.

If rule [P-SYNCH] was applied then

$$\rho \xrightarrow{\alpha} \rho', \quad \sigma \xrightarrow{\beta} \sigma', \quad \alpha \bowtie_c \beta.$$

Since, ρ performs an observable action part (iii) of Lemma 4.2 implies $\rho \stackrel{\checkmark}{\not\longrightarrow}$.

Let us turn our attention to $\text{UNFOLD}(\sigma_1)$. The assumption $\rho \dashv \sigma_1$ together with Proposition 3.23 implies that (a) $\rho \dashv \text{UNFOLD}(\sigma_1)$. The assumption $\sigma_1 \preccurlyeq^{syn}_{\text{SRV}} \sigma$ and Definition 4.8 imply that (b) $\text{UNFOLD}(\sigma_1) \preccurlyeq^{syn}_{\text{SRV}} \sigma$.

We know that $\rho \dashv \text{UNFOLD}(\sigma_1)$ ((a) above), and that $\rho \stackrel{\checkmark}{\not\longrightarrow}$; together with part i, these facts force $\text{UNFOLD}(\sigma_1)$ to offer an action δ such that $\delta \bowtie_c \alpha$. Thus, for some σ_1',

$$\text{UNFOLD}(\sigma_1) \xrightarrow{\delta} \sigma_1'.$$

An application of rule [P-SYNCH] ensures that

$$\rho \parallel \text{UNFOLD}(\sigma_1) \xrightarrow{\tau} \rho' \parallel \sigma_1'.$$

Now (a) and part ii of Definition 3.17 imply that $\rho' \dashv \sigma_1'$. We choose σ_1' as candidate $\hat{\sigma}$.

To finish the proof we have to show that $\sigma_1' \preccurlyeq^{syn}_{\text{SRV}} \sigma'$. The argument is a case analysis on the action β. Four cases are to be discussed, but, as they are all similar, we give a detailed account only of two of them.

If $\beta = ?\texttt{t}_2$ then (b) above and case (ii) of Definition 4.8 ensure that σ' is unique, and so is σ_1' as well. The same definition implies also that $\sigma_1' \preccurlyeq^{syn}_{\text{SRV}} \sigma'$.

If, for some label $\texttt{l}$, $\beta = ?\texttt{l}$ then the definition of $\bowtie_c$ implies that $\alpha = !\texttt{l}$, and the assumption $\delta \bowtie_c \alpha$ implies $\delta = ?\texttt{l}$. We have proven that $\delta = \beta$. Now (b) above and case (iv) of Definition 4.8 imply that $\sigma_1' \preccurlyeq^{syn}_{\text{SRV}} \sigma'$. □

We conclude this sub-section with a summary of our knowledge on the pre-orders which compare contracts on the server side of the compliance relation.

Corollary 4.15. The following equalities and inequalities hold

$$\begin{array}{rcl} \sqsubseteq_{\text{MUST}} & = & \sqsubseteq_{\text{SRV}}. \\ \sqsubseteq^{\mathcal{SC}}_{\text{SRV}} & \not\subseteq & \sqsubseteq_{\text{SRV}}. \\ \sqsubseteq_{\text{SRV}} & \not\subseteq & \sqsubseteq^{\mathcal{SC}}_{\text{SRV}}. \\ \sqsubseteq^{\mathcal{SC}}_{\text{SRV}} & = & \preccurlyeq^{syn}_{\text{SRV}}. \end{array}$$

Proof. The (in)equalities are consequence respectively of Corollary 3.51, Example 4.4, the fact that $\mathcal{SC} \subset \mathcal{C}$, and Theorem 4.14. □

4.3. *The restricted client pre-order*

We introduce a new pre-order which compares the capacity of clients to be satisfied by servers. The structure of this sub-section is similar to that of the previous one on the restricted server pre-order.

Definition 4.16 (restricted client pre-order).
For $\rho_1, \rho_2 \in \mathcal{SC}$ let $\rho_1 \sqsubseteq^{\mathcal{SC}}_{\text{CLT}} \rho_2$ whenever $\rho_1 \dashv \sigma$ implies $\rho_2 \dashv \sigma$ for every σ in $\mathcal{SC}$.

Also on the restricted client pre-order we can reason modulo unfolding.

Proposition 4.17. For every session contract ρ_1 and ρ_2, $\rho_1 \sqsubseteq^{\mathcal{SC}}_{\text{CLT}} \rho_2$ if and only if $\text{UNFOLD}(\rho_2) \sqsubseteq^{\mathcal{SC}}_{\text{CLT}} \text{UNFOLD}(\rho_2)$.

Proof. Follows from Propositions 3.23 and 3.24. □

Example 4.18. We have shown in Example 4.4 that $?\texttt{latte}.\mathbf{1} \sqsubseteq^{\mathcal{SC}}_{\text{SRV}} ?\texttt{moka}.\mathbf{1}+?\texttt{latte}.\mathbf{1}$. A similar argument, this time applied to client-side session contracts, can be used to show that

$$?\texttt{latte}.\mathbf{1} \sqsubseteq^{\mathcal{SC}}_{\text{CLT}} ?\texttt{moka}.?\texttt{moka}.\mathbf{1}+?\texttt{latte}.\mathbf{1}$$

Similarly to what happens for server contracts, if we turn our attention to general contracts then the session contracts above are no longer related. Let us see why. The client $?\texttt{latte}.\mathbf{1}$ complies with the server $!\texttt{latte}.\mathbf{1} + !\texttt{moka}.\mathbf{1}$, because the action $!\texttt{moka}$ will never be performed by the server. On the other hand

$$?\texttt{moka}.?\texttt{moka}.\mathbf{1} + ?\texttt{latte}.\mathbf{1} \parallel !\texttt{latte}.\mathbf{1} + !\texttt{moka}.\mathbf{1} \xrightarrow{\tau} ?\texttt{moka}.\mathbf{1} \parallel \mathbf{1} \overset{\tau}{\not\longrightarrow}$$

and $?\texttt{moka}.\mathbf{1} \overset{\checkmark}{\not\longrightarrow}$; this proves that

$$?\texttt{moka}.?\texttt{moka}.\mathbf{1}+?\texttt{latte}.\mathbf{1} \not\dashv !\texttt{latte}.\mathbf{1} + !\texttt{moka}.\mathbf{1}$$

Had we defined in the obvious way the pre-order $\sqsubseteq_{\text{CLT}}$ on contracts, then the argument above would have proven that

$$!\texttt{latte}.\mathbf{1} \not\sqsubseteq_{\text{CLT}} ?\texttt{moka}.?\texttt{moka}.\mathbf{1}+?\texttt{latte}.\mathbf{1} \tag{6}$$

We have therefore shown that

$$\sqsubseteq^{\mathcal{SC}}_{\text{CLT}} \not\subseteq \sqsubseteq_{\text{CLT}} .$$

We have seen in Proposition 4.7 that the session contract $\mathbf{1}$ is a bottom element in the restricted server pre-order. The client pre-order enjoys the dual property.

Proposition 4.19 (top element). The pre-order $\sqsubseteq^{\mathcal{SC}}_{\text{CLT}}$ enjoys the following two properties,

i. it has a top element
ii. if $\sigma_\top$ is a top element of $\sqsubseteq^{\mathcal{SC}}_{\text{CLT}}$ then $\text{UNFOLD}(\sigma_\top) = \mathbf{1}$.

Proof. Since $\mathbf{1} \dashv \sigma$ for every contract σ, the session contract $\mathbf{1}$ it is a top element in the restricted client pre-order. Moreover, reasoning as in Proposition 4.7 we can show that if $\sigma_\top$ is an arbitrary top element then $\text{UNFOLD}(\sigma_\top) = \mathbf{1}$. □

Definition 4.20 (syntactic sub-client relation).
Let $\mathcal{F}_{\preccurlyeq^{syn}_{\text{CLT}}} : \mathcal{P}(\mathcal{SC}^2) \longrightarrow \mathcal{P}(\mathcal{SC}^2)$ be defined so that $(\rho_1, \rho_2) \in \mathcal{F}_{\preccurlyeq^{syn}_{\text{CLT}}}(\mathcal{R})$ whenever one of the following is true:

i. $\text{UNFOLD}(\rho_2) = \mathbf{1}$.

ii. $\text{UNFOLD}(\rho_2) = ?\mathtt{t}_2.\rho'_2$ and $\text{UNFOLD}(\rho_1) = ?\mathtt{t}_1.\rho'_1$ with $\mathtt{t}_1 \preccurlyeq_{\mathtt{g}} \mathtt{t}_2$ and $\rho'_1 \mathrel{\mathcal{R}} \rho'_2$.
iii. $\text{UNFOLD}(\rho_2) = !\mathtt{t}_2.\rho'_2$ and $\text{UNFOLD}(\rho_1) = !\mathtt{t}_1.\rho'_1$ with $\mathtt{t}_2 \preccurlyeq_{\mathtt{g}} \mathtt{t}_1$ and $\rho'_1 \mathrel{\mathcal{R}} \rho'_2$.
iv. $\text{UNFOLD}(\rho_2) = \sum_{j\in J} \mathtt{l}_j.\rho^2_j$ and $\text{UNFOLD}(\rho_1) = \sum_{i\in I} \mathtt{l}_i.\rho^1_i$ with $I \subseteq J$ and $\rho^1_i \mathrel{\mathcal{R}} \rho^2_i$.
v. $\text{UNFOLD}(\rho_2) = \bigoplus_{j\in J} \mathtt{l}_j.\sigma^2_j$ and $\text{UNFOLD}(\rho_1) = \bigoplus_{i\in I} \mathtt{l}_i.\sigma^1_i$ with $J \subseteq I$ and $\sigma^1_j \mathrel{\mathcal{R}} \sigma^2_j$.

If $\mathcal{R} \subseteq \mathcal{F}_{\preccurlyeq^{syn}_{\text{CLT}}}.(\mathcal{R})$ then we say that $\mathcal{R}$ is a co-inductive syntactic sub-client relation. Let $\preccurlyeq^{syn}_{\text{CLT}}$ denote the greatest solution of the equation $X = \mathcal{F}_{\preccurlyeq^{syn}_{\text{CLT}}}.(X)$; formally,

$$\preccurlyeq^{syn}_{\text{CLT}} \; = \nu\mathcal{F}_{\preccurlyeq^{syn}_{\text{CLT}}}.$$

We call $\preccurlyeq^{syn}_{\text{CLT}}$ the sub-client relation.

Lemma 4.21. Let $\rho_1, \rho_2 \in \mathcal{SC}$, $\rho_1 = \text{UNFOLD}(\rho_1)$, $\rho_2 = \text{UNFOLD}(\rho_2)$ and $\rho_1 \sqsubseteq^{\mathcal{SC}}_{\text{CLT}} \rho_2$. Then

a. if $\rho_2 = !\mathtt{t}_2.\rho'_2$ then $\rho_1 = !\mathtt{t}_1.\rho'_1$, $\mathtt{t}_2 \preccurlyeq_{\mathtt{g}} \mathtt{t}_1$ and $\rho'_1 \sqsubseteq^{\mathcal{SC}}_{\text{CLT}} \rho'_2$
b. if $\rho_2 = ?\mathtt{t}_2.\rho'_2$ then $\rho_1 = ?\mathtt{t}_1.\rho'_1$, $\mathtt{t}_1 \preccurlyeq_{\mathtt{g}} \mathtt{t}_2$ and $\rho'_1 \sqsubseteq^{\mathcal{SC}}_{\text{CLT}} \rho'_2$
c. if $\rho_2 = \sum_{j\in J} ?\mathtt{l}_\mathtt{j}.\rho^2_j$ then $\rho_1 = \sum_{i\in I} ?\mathtt{l}_\mathtt{i}.\rho^1_i$ with $I \subseteq J$ and $\rho^1_i \sqsubseteq^{\mathcal{SC}}_{\text{CLT}} \rho^2_i$
d. if $\rho_2 = \bigoplus_{j\in J} !\mathtt{l}_\mathtt{j}.\rho^2_j$ then $\rho_1 = \bigoplus_{i\in I} !\mathtt{l}_\mathtt{i}.\rho^1_i$ with $J \subseteq I$ and $\rho^1_j \sqsubseteq^{\mathcal{SC}}_{\text{CLT}} \rho^2_j$.

Proof. The proof is almost the same as Lemma 4.9, the difference being that here we look at left-hand side of the compliance relation, and that we need a result of the fact that for each session contract ρ there exists a session contract σ such that $\rho \dashv \sigma$. This is proven in Barbanera and de'Liguoro (2010, Section 2.1). □

Proposition 4.22. For every session contract ρ_1 and ρ_2, if $\rho_1 \sqsubseteq^{\mathcal{SC}}_{\text{CLT}} \rho_2$ then $\rho_1 \preccurlyeq^{syn}_{\text{CLT}} \rho_2$.

Proof. The argument is similar to the one of Proposition 4.10, but here we use the function $\mathcal{F}_{\preccurlyeq^{syn}_{\text{CLT}}}$ and Lemma 4.21. □

Lemma 4.23. Let $\mathcal{R}$ be a co-inductive sub-client relation and let $\rho_1 \preccurlyeq^{syn}_{\text{CLT}} \rho_2$. If $\rho_2 \xrightarrow{\tau} \rho'_2$ then $\rho_1 \preccurlyeq^{syn}_{\text{CLT}} \rho'_2$.

Proof. The proof is similar to the proof of Lemma 4.11. □

Theorem 4.24 (co-inductive characterization). Let $\rho, \sigma \in \mathcal{SC}$. Then $\rho \preccurlyeq^{syn}_{\text{CLT}} \sigma$ if and only if $\rho \sqsubseteq^{\mathcal{SC}}_{\text{CLT}} \sigma$.

Proof. In view of Proposition 4.22 we have to prove only the inclusion $\preccurlyeq^{syn}_{\text{CLT}} \subseteq \sqsubseteq^{\mathcal{SC}}_{\text{CLT}}$. It is enough to show that

$$\mathcal{R} = \{\, (\rho_2, \sigma) \mid \rho_1 \preccurlyeq^{syn}_{\text{CLT}} \rho_2,\ \rho_1 \dashv \sigma \ for\ some\ \rho_1 \in \mathcal{SC} \,\}$$

is a co-inductive compliance. Let $\rho \mathrel{\mathcal{R}} \sigma$; by definition of $\mathcal{R}$ there exists a session contract ρ_1 such that $\rho_1 \preccurlyeq^{syn}_{\text{CLT}} \rho$ and $\rho_1 \dashv \sigma$.

We prove part i of Definition 3.17. Assume $\rho \parallel \sigma \stackrel{\tau}{\not\longrightarrow}$; we have to show that $\rho \xrightarrow{\checkmark}$. To this aim it is sufficient to show

$$\text{UNFOLD}(\rho_1) = \mathbf{1}. \tag{7}$$

We explain why this fact suffices. Assume (7). Since $\text{UNFOLD}(\rho_1) \preccurlyeq^{syn}_{\text{CLT}} \rho$ we know that

$$(\rho_1, \rho) \in \mathcal{F}_{\preccurlyeq^{syn}_{\text{CLT}}}(\preccurlyeq^{syn}_{\text{CLT}}).$$

This is possible only thanks to case (i) of Definition 4.20, and therefore $\text{UNFOLD}(\rho) = \mathbf{1}$. Since $\rho \stackrel{\tau}{\not\longrightarrow}$, part (i) of Lemma 3.2 implies $\rho = \text{UNFOLD}(\rho)$, and so, now, an application of part (ii) of Lemma 4.2 ensures $\rho \stackrel{\checkmark}{\longrightarrow}$.

We prove (7). The argument revolves around the unfolding of ρ_1. To begin with, note two facts: one, the assumption $\rho_1 \preccurlyeq^{syn}_{\text{CLT}} \rho$ and Definition 4.20 imply $\text{UNFOLD}(\rho_1) \preccurlyeq^{syn}_{\text{CLT}} \rho$; and the other, the assumption $\rho_1 \dashv \sigma$ and Proposition 3.23 imply $\text{UNFOLD}(\rho_1) \dashv \sigma$.

The fact that $\rho \parallel \sigma \stackrel{\tau}{\not\longrightarrow}$ can be used to prove

$$\alpha \in \text{R} \text{ implies } \alpha \not\bowtie_c \beta \text{ for every } \beta \in \text{S}. \tag{8}$$

From the definition of acceptance set and $\rho_1 \stackrel{\tau}{\longrightarrow}{}^{*} \text{UNFOLD}(\rho_1)$ (part ii of Lemma 3.2) it follows

$$\text{Acc}(\text{UNFOLD}(\rho_1)) \subseteq \text{Acc}(\rho_1). \tag{9}$$

Now we prove that $\text{UNFOLD}(\rho_1) = \mathbf{1}$. Fix a *stuck* derivative ρ_1' of $\text{UNFOLD}(\rho_1)$:

$$\text{UNFOLD}(\rho_1) \stackrel{\tau}{\longrightarrow}{}^{*} \rho_1' \stackrel{\tau}{\not\longrightarrow}.$$

Such a stuck state exists because of the restricted syntax of session contracts. Further, since $\rho_1 \stackrel{\tau}{\not\longrightarrow}$, by definition we have $\rho_1 \downarrow \text{R}$ for some R. Point (9) implies that $\text{R} \in \text{Acc}(\rho_1)$, and so point (8), together with $\rho_1 \stackrel{\tau}{\not\longrightarrow}$ and $\sigma \stackrel{\tau}{\not\longrightarrow}$, implies that $\rho_1' \parallel \sigma \stackrel{\tau}{\not\longrightarrow}$. We have already seen that $\text{UNFOLD}(\rho_1) \dashv \sigma$, so part ii of Definition 3.17 now imply that $\rho_1' \stackrel{\checkmark}{\longrightarrow}$. We can now apply part (ii) of Lemma 4.2 to obtain $\text{UNFOLD}(\rho_1) = \mathbf{1}$.

As yet we have proven that (ρ, σ) respects part i of Definition 3.17. The argument to show that also part ii of Definition 3.17 holds is similar to the one used in Theorem 4.14. The difference amounts in the use of Definition 4.20 in place of Definition 4.8. We leave the details to the reader. □

5. Modelling session types

The interpretation of session types as contracts is expressed as a function from the language $L_{\mathcal{ST}}$ in Section 2 to the language $L_{\mathcal{SC}}$ in Section 4. The function is little more than a syntactic transformation.

Let $\mathcal{M} : L_{\mathcal{ST}} \longrightarrow L_{\mathcal{SC}}$ be defined by:

$$\mathcal{M}(S) = \begin{cases} \mathbf{1} & \text{if } S = \text{END} \\ !\texttt{t}.\mathcal{M}(S) & \text{if } S = ![\texttt{t}];S \\ ?\texttt{t}.\mathcal{M}(S) & \text{if } S = ?[\texttt{t}];S \\ \sum_{i\in[1;n]} ?\texttt{l}_i.\mathcal{M}(S_i) & \text{if } S = \&\langle\, \texttt{l}_1 : S_1, \ldots, \texttt{l}_n : S_n \,\rangle \\ \bigoplus_{i\in[1;n]} !\texttt{l}_i.\mathcal{M}(S_i) & \text{if } S = \oplus\langle\, \texttt{l}_1 : S_1, \ldots, \texttt{l}_n : S_n \,\rangle \\ \mu x.\, \mathcal{M}(S') & \text{if } S = \mu X.\, S' \\ x & \text{if } S = X \end{cases}$$

It is easy to see that $\mathcal{M}$ maps session types, $\mathcal{ST}$, to session contracts, $\mathcal{SC}$; indeed it defines a bijection between these sets:

— for every $\sigma \in \mathcal{SC}$ there exists some session type T such that $\mathcal{M}(T) = \sigma$
— if $\mathcal{M}(T_1) = \mathcal{M}(T_2)$ then $T_1 = T_2$

where $T_1 = T_2$ denotes syntactic identity. Further, substitution is preserved by $\mathcal{M}$.

Lemma 5.1. Let $S, T \in \mathcal{ST}$. Then $\mathcal{M}(S\{^T/_X\}) = (\mathcal{M}(S))\{^{\mathcal{M}(T)}/_{\mathcal{M}(X)}\}$.

Proof. The proof is by structural induction on S. □

The interpretation also commutes with the two functions $depth(-)$ and $\text{UNFOLD}(-)$:

Lemma 5.2. For every $T \in \mathcal{ST}$ and $\sigma \in \mathcal{SC}$ the ensuing properties are true,

i. $dpt(T) = dpt(\mathcal{M}(T))$.
ii. $\text{UNFOLD}(\mathcal{M}(T)) = \mathcal{M}(\text{UNFOLD}(T))$.
iii. $\text{UNFOLD}(\mathcal{M}^{-1}(\sigma)) = T$ if and only if $\text{UNFOLD}(\sigma) = \mathcal{M}(T)$.

Proof. The proofs of the first two points are by induction on $dpt(T)$, the proof of (ii) using (i) and the previous lemma. The third point follows immediately from (ii). □

As we have shown, the difficulty is to find a natural pre-order on session contracts which accurately reflects the sub-typing relation on session contracts. There are two obvious candidates, the restricted server pre-order and the restricted client pre-order on session contracts. The difficulty lies in the interpretation of END.

Example 5.3. Recall that $\mathcal{M}(\text{END}) = \mathbf{1}$. In the restricted server pre-order the session contract $\mathbf{1}$ is a least element, being smaller or equal to every other session contract. On the other hand, for session types $\text{END} \preccurlyeq_{\text{ST}} T$ if and only if $\text{UNFOLD}(T) = \text{END}$. Consequently the relation $\sqsubseteq^{\mathcal{SC}}_{\text{SRV}}$ is an unsound model for sub-typing between session types. For example:

$$\mathbf{1} \sqsubseteq^{\mathcal{SC}}_{\text{SRV}} !\mathtt{t}.\mathbf{1}, \quad \text{END} \not\preccurlyeq_{\text{ST}} ![\mathtt{t}];\text{END}.$$

The restricted client pre-order presents the dual issue as it relates every session contract to $\mathbf{1}$; it is one of the top element. Once again a model based on $\sqsubseteq^{\mathcal{SC}}_{\text{CLT}}$ would be unsound:

$$!\mathtt{t}.\mathbf{1} \sqsubseteq^{\mathcal{SC}}_{\text{CLT}} \mathbf{1}, \quad ![\mathtt{t}];\text{END} \not\preccurlyeq_{\text{ST}} \text{END}.$$

The main result of the paper is that the bijection $\mathcal{M}$ gives a fully abstract interpretation of sub-typing between session types in terms of session contracts, provided we combine these two set-based pre-orders.

Definition 5.4 (session contract pre-order). For $\sigma_1, \sigma_2 \in \mathcal{SC}$ let $\sigma_1 \sqsubseteq^{\mathcal{SC}} \sigma_2$ whenever $\sigma_1 \sqsubseteq^{\mathcal{SC}}_{\text{SRV}} \sigma_2$ and $\sigma_1 \sqsubseteq^{\mathcal{SC}}_{\text{CLT}} \sigma_2$.

Example 5.5. It is instructive to see the behaviour of $\mathbf{1}$, the image of END under $\mathcal{M}$, relative to this combined pre-order. First suppose $\sigma \sqsubseteq^{\mathcal{SC}} \mathbf{1}$ for some session contract σ. This implies $\sigma \sqsubseteq^{\mathcal{SC}}_{\text{SRV}} \mathbf{1}$ and therefore, as we have shown in Proposition 4.7, σ must be a bottom element relative to $\sqsubseteq^{\mathcal{SC}}_{\text{SRV}}$ and $\text{UNFOLD}(\sigma)$ must be $\mathbf{1}$. A similar argument, using the pre-order $\sqsubseteq^{\mathcal{SC}}_{\text{CLT}}$ ensures that if $\mathbf{1} \sqsubseteq^{\mathcal{SC}} \sigma$ then $\text{UNFOLD}(\sigma)$ must also be $\mathbf{1}$.

In other words, modulo unfolding the only session contract related to $\mathbf{1}$ via $\sqsubseteq^{\mathcal{SC}}$ is $\mathbf{1}$ itself.

Proposition 5.6 (completeness). For session contracts, $\sigma_1 \sqsubseteq^{\mathcal{SC}} \sigma_2$ implies $\mathcal{M}^{-1}(\sigma_1) \preccurlyeq_{\mathrm{ST}} \mathcal{M}^{-1}(\sigma_2)$.

Proof. Let $\mathcal{R}$ be the relation over session types defined by

$$\mathcal{R}= \{(\mathcal{M}^{-1}(\sigma_1), \mathcal{M}^{-1}(\sigma_2)) \mid \sigma_1 \preccurlyeq^{syn}_{\mathrm{SRV}} \sigma_2,\ \sigma_1 \preccurlyeq^{syn}_{\mathrm{CLT}} \sigma_2\}.$$

If we prove that $\mathcal{R}$ is a type simulation, that is it satisfies the properties given in Definition 2.10, then the result will follow because of Theorems 4.14 and 4.24.

The proof proceeds by a case analysis on the structure of $\textsc{unfold}(\sigma_1)$; we give the details of two cases.

— Suppose $\textsc{unfold}(\mathcal{M}^{-1}(\sigma_1)) = \textsc{end}$. According to Definition 2.10 we have to show that

$$\textsc{unfold}(\mathcal{M}^{-1}(\sigma)) = \textsc{end}.$$

Because of part (iii) of Lemma 5.2 we know that $\textsc{unfold}(\sigma_1) = \mathbf{1}$; moreover in Example 5.5 above we have already reasoned that $\textsc{unfold}(\sigma_2)$ must be $\mathbf{1}$.

— Suppose $\textsc{unfold}(\mathcal{M}^{-1}(\sigma_1)) = ![\mathtt{t}_1]\,;S_1$. We are required to prove that

$$\textsc{unfold}(\mathcal{M}^{-1}(\sigma_2)) = ![\mathtt{t}_2]\,;S_2, \tag{10}$$

for some $\mathtt{t}_2$ and S_2 such that $\mathtt{t}_2 \preccurlyeq_{\mathtt{g}} \mathtt{t}_1$ and $(\mathcal{M}(S_1), \mathcal{M}(S_2)) \in \preccurlyeq^{syn}_{\mathrm{SRV}} \cap \preccurlyeq^{syn}_{\mathrm{CLT}}$.

Again by Lemma 5.2 (iii) we know that $\textsc{unfold}(\sigma_1) = !\mathtt{t}_1.\mathcal{M}(S_1)$. As $\sigma_1 \preccurlyeq^{syn}_{\mathrm{SRV}} \sigma_2$, Definition 4.8 implies that $\textsc{unfold}(\sigma_2) = !\mathtt{t}_2.\sigma_2'$ for some $\mathtt{t}_2$ such that $\mathtt{t}_2 \preccurlyeq_{\mathtt{g}} \mathtt{t}_1$ and some σ_2' such that $\mathcal{M}(S_1) \preccurlyeq^{syn}_{\mathrm{SRV}} \sigma_2'$. Now letting S_2 denote $\mathcal{M}^{-1}(\sigma_2')$, another application of Lemma 5.2 (iii) ensures that (10) above is satisfied. By the definition of S_2 we also have the requirement $\mathcal{M}(S_1) \preccurlyeq^{syn}_{\mathrm{SRV}} \mathcal{M}(S_2)$.

It remains to show $\mathcal{M}(S_1) \preccurlyeq^{syn}_{\mathrm{CLT}} \mathcal{M}(S_2)$. But this follows from $\sigma_1 \preccurlyeq^{syn}_{\mathrm{CLT}} \sigma_2$, by part (iii) of Definition 4.20.

The proof for the remaining cases is similar to the argument already shown, and left to the reader. □

Theorem 5.7 (full abstraction).
For all session types, $T_1 \preccurlyeq_{\mathrm{ST}} T_2$ if and only if $\mathcal{M}(T_1) \sqsubseteq^{\mathcal{SC}} \mathcal{M}(T_2)$.

Proof. Thanks to the completeness theorem, Theorem 5.6, it is sufficient to prove that $T_1 \preccurlyeq_{\mathrm{ST}} T_2$ implies $\mathcal{M}(T_1) \sqsubseteq^{\mathcal{SC}}_{\mathrm{SRV}} \mathcal{M}(T_2)$ and $\mathcal{M}(T_1) \sqsubseteq^{\mathcal{SC}}_{\mathrm{CLT}} \mathcal{M}(T_2)$. As an example we outline the proof of the former. Because of Theorem 4.14 it is sufficient to show that the relation $\mathcal{R}$ given by

$$\mathcal{R} = \{\, (\sigma_1, \sigma_2) \mid \mathcal{M}^{-1}(\sigma_1) \preccurlyeq_{\mathrm{ST}} \mathcal{M}^{-1}(\sigma_2)\,\}$$

is a syntactic sub-server relation, that is $\mathcal{R} \subseteq \mathcal{F}_{\preccurlyeq^{syn}_{\mathrm{SRV}}}(\mathcal{R})$, where $\mathcal{F}_{\preccurlyeq^{syn}_{\mathrm{SRV}}}$ is given in Definition 4.8.

Suppose $\sigma_1 \mathrel{\mathcal{R}} \sigma_2$. The proof is a case analysis.

— If $\textsc{unfold}(\sigma_1) = \mathbf{1}$ we have nothing to prove because condition (i) of Definition 4.8 does not require anything.

— If $\text{UNFOLD}(\sigma_1) = ?\mathtt{t}_1.\sigma_1'$ we have to show that

$$\text{UNFOLD}(\sigma_2) = ?\mathtt{t}_2.\sigma_2'$$

with $\mathtt{t}_1 \leqslant_{\mathsf{g}} \mathtt{t}_2$ and $\sigma_1' \mathcal{R} \sigma_2'$. An application of part (iii) of Lemma 5.2 shows that $\text{UNFOLD}(\mathcal{M}^{-1}(\sigma_1)) = ?[\mathtt{t}_1]; \mathcal{M}^{-1}(\sigma_1')$. The fact that $\mathcal{M}^{-1}(\sigma_1) \leqslant_{\text{ST}} \mathcal{M}^{-1}(\sigma_2)$ let us use Definition 2.10 to deduce that $\mathcal{M}^{-1}(\sigma_2) = ?[\mathtt{t}_2]; \mathcal{M}^{-1}(\sigma_2')$ for some $\mathtt{t}_2$ such that $\mathtt{t}_1 \leqslant_{\mathsf{g}} \mathtt{t}_2$, and some $\mathcal{M}^{-1}(\sigma_2')$ such that $\mathcal{M}^{-1}(\sigma_1') \leqslant_{\text{ST}} \mathcal{M}^{-1}(\sigma_2')$. From the last inequality and the definition of $\mathcal{R}$ it follows that $\sigma_1' \mathcal{R} \sigma_2'$. Since we have proven the conditions on the input actions $\mathtt{t}_1$, $\mathtt{t}_2$ and on the continuations σ_1', σ_2' we have left only to show that the structure of $\text{UNFOLD}(\sigma_2)$ is the required one; this follows from another application of part (iii) of Lemma 5.2.

The other cases are analogous and left to the reader. □

Corollary 5.8. The relation $\sqsubseteq^{\mathcal{SC}}$ is decidable.

Proof. To begin with, note that $\mathcal{M}$ is defined by structural induction, so it is decidable. The corollary then follows from Corollary 2 of Gay and Hole (2005), which ensures that the relation $\leqslant_{\text{ST}}$ is decidable, and our Theorem 5.7, whereby we can prove the isomorphism $\leqslant_{\text{ST}} \cong \sqsubseteq^{\mathcal{SC}}$. □

5.1. *Examples and applications*

In this sub-section we give a series of examples in order to discuss the results we obtained. The first two example are of theoretical nature, whereas the last one shows an application.

Example 5.9 (type simulations and the weak simulation relation). At this stage, a natural question arises, which concerns the relationship between type simulations and weak simulations (Milner 1999). Assume the standard definition of the weak simulation (Milner 1999); we use the symbol $\lesssim$ to denote the greatest weak simulation relation.

We begin by showing that, even though two session types are in a co-inductive types simulation, their images through $\mathcal{M}$ need not be in a weak simulation. Consider the relation

$$\mathcal{R} = \{(\oplus\langle\, \mathtt{l}_1 : \text{END},\ \mathtt{l}_2 : \text{END}\,\rangle, (\oplus\langle\, \mathtt{l}_1 : \text{END}\,\rangle), (\text{END}, \text{END})\}.$$

The standard co-inductive proof technique let one prove that the relation $\mathcal{R}$ is a type simulation. On the other hand, the definition of $\mathcal{M}$ implies that

$$\mathcal{M}(\oplus\langle\, \mathtt{l}_1 : \text{END},\ \mathtt{l}_2 : \text{END}\,\rangle) = !\mathtt{l}_1.\mathbf{1} \oplus !\mathtt{l}_2.\mathbf{1}$$
$$\mathcal{M}(\oplus\langle\, \mathtt{l}_1 : \text{END}\,\rangle) = !\mathtt{l}_1.\mathbf{1}$$

Then $\mathcal{M}(\oplus\langle\, \mathtt{l}_1 : \text{END},\ \mathtt{l}_2 : \text{END}\,\rangle) \not\lesssim \mathcal{M}(\oplus\langle\, \mathtt{l}_1 : \text{END}\,\rangle)$ because $!\mathtt{l}_1.\mathbf{1} \oplus !\mathtt{l}_2.\mathbf{1} \xrightarrow{\tau}\xrightarrow{\mathtt{l}_2}$, while $!\mathtt{l}_1.\mathbf{1} \stackrel{\mathtt{l}_2}{\not\longrightarrow}{}^{*}$. We have proven that $S_1 \leqslant_{\text{ST}} S_2$ does not imply $\mathcal{M}(S_1) \lesssim \mathcal{M}(S_2)$.

Looking at the foregoing argument, one might be tempted to reason that if $S_1 \leqslant_{\text{ST}} S_2$ then $\mathcal{M}(S_2) \lesssim \mathcal{M}(S_1)$. We prove that this is not the case. We can prove that

$$?\mathtt{l}_1.\mathbf{1} \sqsubseteq^{\mathcal{SC}} ?\mathtt{l}_2.\mathbf{1} + ?\mathtt{l}_1.\mathbf{1}$$

An application of $\mathcal{M}^{-1}$ gives us:

$$\mathcal{M}^{-1}(?\mathtt{l}_1.\mathbf{1}) = \&\langle\, \mathtt{l}_1 : \textsc{end} \,\rangle$$
$$\mathcal{M}^{-1}(?\mathtt{l}_2.\mathbf{1}+?\mathtt{l}_1.\mathbf{1}) = \&\langle\, \mathtt{l}_1 : \textsc{end},\ \mathtt{l}_2 : \textsc{end} \,\rangle.$$

A look at the definition of $\preccurlyeq_{\textsc{st}}$, Definition 2.10, lets one prove that for every type simulation $\mathcal{R}$

$$(\&\langle\, \mathtt{l}_1 : \textsc{end},\ \mathtt{l}_2 : \textsc{end} \,\rangle,\ \&\langle\, \mathtt{l}_1 : \textsc{end} \,\rangle) \notin \mathcal{R}$$

and, therefore,

$$\&\langle\, \mathtt{l}_1 : \textsc{end},\ \mathtt{l}_2 : \textsc{end} \,\rangle \not\preccurlyeq_{\textsc{st}} \&\langle\, \mathtt{l}_1 : \textsc{end} \,\rangle.$$

Example 5.10 (e-vote, revisited).
In this example we use Theorem 5.7 in conjunction with Theorem 2 of Gay and Hole (2005), in order to show how the set based pre-order $\sqsubseteq^{\mathcal{SC}}$ can be used to guarantee that a process P_a can be safely replaced by a suitable process P_b.

Consider two contracts BallotA and BallotB such that BallotA $\sqsubseteq^{\mathcal{SC}}$ BallotB. Let $BallotA = \mathcal{M}^{-1}(\text{BallotA})$ and $BallotB = \mathcal{M}^{-1}(\text{BallotB})$. From Theorem 5.7 it follows that

$$BallotA \preccurlyeq_{\textsc{st}} BallotB. \tag{11}$$

Let $\perp_c$ denote the *coinductive duality relation* defined as in Definition 9 of Gay and Hole (2005). Suppose now that $\textsc{BltSrvA}(x^+)$, $\textsc{BltSrvB}(x^+)$ and $\textsc{Voter}(x^-)$ are pi calculus processes (as in Gay and Hole (2005)) such that

$$\{x^+ : BallotA\} \vdash \textsc{BltSrvA}(x^+),$$
$$\{x^+ : BallotB\} \vdash \textsc{BltSrvB}(x^+),$$
$$\{x^- : Voter\} \vdash \textsc{Voter}(x^-)$$

for some session type $Voter$ such that $Voter \perp_c BallotA$. By means of the typing rules of Gay and Hole (2005), it is possible to derive

$$\dfrac{\dfrac{\dfrac{\vdots}{\{x^+ : BallotA\} \vdash \textsc{BltSrvA}(x^+)} \quad \dfrac{\vdots}{\{x^- : Voter\} \vdash \textsc{Voter}(x^-)}}{\{x^+ : BallotA\}, x^- : Voter \vdash \textsc{BltSrvA}(x^+) \mid \textsc{Voter}(x^-)}\,[\text{T-Par}]}{\vdash (\nu x : BallotA)\ \textsc{BltSrvA}(x^+) \mid \textsc{Voter}(x^-)}\,[\text{T-NewS}]$$

Then (11) above and Theorem 2 of Gay and Hole (2005) can be used to guarantee that if process $\textsc{BltSrvB}(x^+)$ is used in place of process $\textsc{BltSrvA}(x^+)$, then no communication error will happen along the channel x.

One can use non-recursive versions of the contracts seen in Examples 3.14 and 4.5 to obtain contracts that satisfy the assumptions above:

$$\begin{aligned}
\text{BallotA} &= ?\mathtt{Login}.(\ !\mathtt{Wrong}.\mathbf{1} \oplus !\mathtt{Ok}.(\ ?\mathtt{VoteA}.\mathbf{1} + ?\mathtt{VoteB}.\mathbf{1}\,)\,) \\
\text{BallotB} &= ?\mathtt{Login}.(\ !\mathtt{Wrong}.\mathbf{1} \oplus \\
&\qquad\qquad !\mathtt{Ok}.(\ ?\mathtt{VoteA}.\mathbf{1} + ?\mathtt{VoteB}.\mathbf{1} + ?\mathtt{VoteC}.\mathbf{1} + ?\mathtt{VoteD}.\mathbf{1}\,)\,) \\
Voter &= \mathcal{M}^{-1}(!\mathtt{Login}.(\ ?\mathtt{Wrong}.\mathbf{1} + ?\mathtt{Ok}.(\ !\mathtt{VoteA}.\mathbf{1} \oplus !\mathtt{VoteB}.\mathbf{1}\,)\,))
\end{aligned}$$

Example 5.11 (protocol conformance). As already remarked, the language for contracts is a sub-language of CCS without τ's (Nicola and Hennessy 1987), and consequently contracts are suitable for specifying communication protocols.

Assume a protocol Pr to be specified by a contract σ, and let Q be a process (in the sense of Gay and Hole (2005)), which is well typed under the environment Γ. Assume also that $\Gamma(x) = S$ for some channel x.

We want to answer the following question:

(Q) 'Does the session type S conform to the protocol specification σ?'

Clearly, as long as the notion of conformance is not mathematically defined, it is not possible to give an answer (at least not a meaningful one).

In light of Theorem 5.7, we propose the following definition of conformance. Assume the standard definition of weak bisimilarity equivalence (Milner 1999); we denote this relation $\approx$. We say that a session type S conforms to a protocol specification σ if and only if $\mathcal{M}(S) \approx \sigma$.

To answer the question (Q) now one has only to prove that $\mathcal{M}(S) \approx \sigma$ or to show a counter example to this statement.

For example, if we had given a specification of the protocol POP3 (Rose 1988) with a contract σ, then we would have been able to check whether the session type POP3 of Gay *et al.* (2003) conforms to σ.

In order for the notion of conformance we have given to be of any practical consequence, one last thing has to be ascertained. We have to prove that weak bisimilarity equivalence, when restricted to session contracts, is *decidable*. We leave this as an open problem worth further investigation.

6. Conclusions

6.1. *Summary*

In this paper, we have used contracts (Padovani 2010) to give a fully abstract model for first-order session types ordered by their sub-typing relation (Gay and Hole 2005).

In view of the interpretation $\mathcal{M}$, we have shown that the 'natural' server refinement on contract (Definition 3.25) is not a *complete* model for the sub-typing, as it does not allow the refinement $a \sqsubseteq a + b$. Example 4.18 shows that also the 'natural' client pre-order on contracts has the same issue. This has lead us to the identification of a subset of contracts which we call session contracts. We have then shown that the 'natural' server and client refinements on session contracts are *unsound* models of the sub-typing (Example 5.3). This result explains why to obtain a fully-abstract model, it is *necessary* to introduce yet another pre-order, that is the intersection of the server and the client pre-orders on session contracts (Definition 5.4).

We believe that our work

— provides the first fully-abstract model of session types in terms of contracts;
— shows the first alternative characterization of the server and the client pre-orders on session contracts;

Behavioural		Co-inductive
$\sqsubseteq_{\mathrm{SRV}}$	$=$	$\preccurlyeq_{\mathrm{SRV}}$
$\sqsubseteq^{\mathcal{SC}}_{\mathrm{SRV}}$	$=$	$\preccurlyeq^{syn}_{\mathrm{SRV}}$
$\sqsubseteq^{\mathcal{SC}}_{\mathrm{CLT}}$	$=$	$\preccurlyeq^{syn}_{\mathrm{CLT}}$
$\sqsubseteq^{\mathcal{SC}}$	$\cong$	$\preccurlyeq_{\mathrm{ST}}$

Fig. 10. Our behavioural pre-orders for contracts, the relations characterizing them, and the isomorphism with sub-typing.

— contains the first published proof that shows the equality between the must pre-order and a refinement for servers based on the compliance relation; moreover, it shows that the equality holds regardless of the $\bowtie$ relation used to let contracts interact.

The table in Figure 10 sketches our knowledge on the refinements that we have investigated. By means of examples we have shown that the relations in different rows are different.

If we bear in mind the debate on the 'right' notion of compliance for contracts, Section 3.3 and Corollary 3.51 are worthy of comment. The result states that as long as we are concerned with refinements for the servers, it does not matter whether we pick as definition of compliance Definition 3.17 or Definition 3.40, because the resulting refinements are the same. This means also that the must pre-order on contracts does not provide a complete model for the sub-typing.

6.2. *Related work*

Roughly speaking, the refinements for contracts that have been proposed thus far in the literature can be related either to the must testing (Nicola and Hennessy 1984), or to the fair testing (Rensink and Vogler 2007). According to this criterion, we divide this section in two parts; first we compare the refinements we have investigated with the pre-orders used in the papers that have influenced us most, namely Barbanera and de'Liguoro (2010) Laneve and Padovani (2007, 2008) and Padovani (2010). The theories presented in these papers are related the must pre-order; thus we will refer to them as *must-theories*. Afterwards, we compare our work with two theories inspired by the fair testing, which are given in Bravetti and Zavattaro (2009) and Padovani (2011). We refer to those theories as *fair-theories*. As we will see, the must-theories bear some similarities with our results; whereas the fair-theories turns out to provide refinements that are different from the ones we have studied.

From now on we reason under the assumption that in our definition of compliance the synchronization relation $\bowtie_i$ is used.

6.2.1. *Must-theories.* The comparison with Laneve and Padovani (2007, 2008) is complicated by the fact that in these papers compliance judgements take the form $\mathrm{I}_1[\rho] \dashv \mathrm{I}_2[\sigma]$ where $\mathrm{I}_1, \mathrm{I}_2$ are finite sets of actions representing in some sense the interfaces of the processes guaranteeing the contracts; moreover, for a contract $\mathrm{I}[\sigma]$ to be valid its interface I

has to contain all the action names (including ✓) that appear in the behaviour σ. Let us refer to the pairs $\mathrm{I}[\sigma]$ as *constrained contracts.*

In Laneve and Padovani (2007), a theorem is proven, which resembles our Corollary 3.51:

Theorem 2. Let $\mathrm{I} = names(\tau)$. $\mathrm{I}[\sigma] \preceq^{lp07} \mathrm{I}[\tau]$ if and only if $\sigma \sqsubseteq_{\text{MUST}} \tau$.

This theorem is weaker than Corollary 3.51. First of all, as constrained contracts are pairs (i.e. a set of actions and a behaviour), the server refinement $\preceq^{lp07}$ (Laneve and Padovani 2007, see Definition 2) *cannot* be directly compared with the must pre-order; formally

$$\preceq^{lp07} \not\subseteq \sqsubseteq_{\text{MUST}}, \quad \sqsubseteq_{\text{MUST}} \not\subseteq \preceq^{lp07} . \tag{12}$$

One may argue that this has no significance, as it is still easy to prove that

$$\text{if } \sigma_1 \sqsubseteq_{\text{MUST}} \sigma_2 \text{ then } names(\sigma_2)[\sigma_1] \preceq^{lp07} names(\sigma_2)[\sigma_2].$$

The converse implication, though, relies heavily on the interfaces of the constrained contracts at hand, and in general is not true:

$$\mathrm{I}[\sigma] \preceq^{07} \mathrm{I'}[\sigma'] \text{ does not imply that } \sigma \sqsubseteq_{\text{MUST}} \sigma'. \tag{13}$$

For instance, we can prove the following

$$\begin{array}{lll} \varnothing[\text{NIL}] & \preceq^{lp07} & \{\,!a\,\}[!a.\text{NIL}] \\ \text{NIL} & \not\sqsubseteq_{\text{MUST}} & !a.\text{NIL} \end{array}$$

It follows that *in the sense shown above* the pre-order $\preceq^{lp07}$ is coarser than the must pre-order.

Since the relation $\preceq^{lp07}$ cannot be compared directly with $\sqsubseteq_{\text{MUST}}$ (12), and is somehow coarser than $\sqsubseteq_{\text{MUST}}$ (13), to the best of our knowledge, our Corollary 3.51 is the first published proof of the equality between a first-order compliance-based refinement and a testing based must pre-order.

Also there is a difference between our compliance relation and the compliance of Laneve and Padovani (2007). The compliance used in Laneve and Padovani (2007), which we denote $\dashv^{lp07}$ is related to our compliance in the sense that

Proposition. If $\mathrm{I}[\rho] \dashv^{lp07} \mathrm{J}[\sigma]$ for some I, J then $\rho \dashv \sigma$.

The converse of the proposition is not true; we explain why. We have $?a.✓.\text{NIL} \dashv !a.✓.\text{NIL}$, while $✓ \in names(!a.✓.\text{NIL})$ thus $?a.✓.\text{NIL} \not\dashv^{lp07} !a.✓.\text{NIL}$, because Definition 1 of Laneve and Padovani (2007) requires that ✓ be not among the names of the contract on the right hand side of the compliance.

Now we compare our work with Laneve and Padovani (2008); we denote $\dashv^{lp08}$ the compliance relation given by Laneve and Padovani (2008, Definition 1), and $\preceq^{lp08}$ the pre-order given there in Definition 2.

Our compliance relation and $\dashv^{lp08}$ are related in a way analogous to what we have seen for $\dashv^{07}$.

Proposition. If $\mathrm{I}[\rho] \dashv^{lp08} \mathrm{J}[\sigma]$ for some I, J then $\rho \dashv \sigma$.

The converse is not true, $?a.\checkmark.\textsc{nil}+?b.\checkmark.\textsc{nil} \dashv !a.\checkmark.\textsc{nil}$, and from $\{a,b\} \not\subseteq \{a\}$ it follows

$$\{a, b, \checkmark\}[?a.\checkmark.\textsc{nil}+?b.\checkmark.\textsc{nil}] \not\dashv^{lp08} \{a, \checkmark\}[!a.\checkmark.\textsc{nil}].$$

We can prove that the relation $\preceq^{lp08}$ extends $\preceq^{07}$ to the terms that contain $\checkmark$; for instance we can prove that

$$\begin{array}{l}\{\checkmark\}[\checkmark.\textsc{nil}] \preceq^{lp08} \emptyset[\textsc{nil}]\\ \{\checkmark\}[\checkmark.\textsc{nil}] \preceq^{lp08} \{\ell, \checkmark\}[\ell.\checkmark.\textsc{nil}].\end{array}$$

Laneve and Padovani (2008) presents the first comparison between session types and contracts; in particular, it is shown that the subcontract relation $\preceq^{lp08}$ together with two interpretations similar to $\mathcal{M}$ provide two sound models for the sub-typing on a subset of our session types. These interpretations are denoted $[\![-]\!]_{\mathbf{1}}$ and $[\![-]\!]_{\textsc{nil}}$. Their proposed full abstraction result, (Laneve and Padovani 2008, see Theorem 2), though, appears not to be true. According to that definition and the interpretation $[\![-]\!]_{\textsc{nil}}$

$$\emptyset[\textsc{nil}] \preceq^{lp08} \{\ell\}[\ell.\textsc{nil}].$$

Their Theorem 2 therefore implies $\&\langle \ell : \textsc{end} \rangle \preccurlyeq_{\textsc{st}} \textsc{end}$, which is not true. On the other hand if $[\![-]\!]_{\mathbf{1}}$ is used then there are two issues. According to Theorem 2 the pair $(\textsc{end}, \&\langle \ell : \textsc{end} \rangle)$ is interpreted as $(\emptyset[\checkmark.\textsc{nil}], \{\ell\}[\ell.\checkmark.\textsc{nil}])$. Then

a. neither $\emptyset[\checkmark.\textsc{nil}]$ nor $\{\ell\}[\ell.\checkmark.\textsc{nil}]$ are constrained contracts, because their interfaces do not contain all the action names which appear in the respective behaviours; moreover
b. even if the interpretation was correct, Theorem 2 would be false because

$$\{\checkmark\}[\checkmark.\textsc{nil}] \preceq^{lp08} \{\ell, \checkmark\}[\ell.\checkmark.\textsc{nil}]$$

while, as stated above, $\&\langle \ell : \textsc{end} \rangle \preccurlyeq_{\textsc{st}} \textsc{end}$ is not true.

Our study of the restricted pre-orders on session contracts (Definitions 4.3 and 4.16) is clearly inspired by Barbanera and de'Liguoro (2010). In Barbanera and de'Liguoro (2010), the language for session types is the same one as we used, whereas the subset of contracts in which session types are embedded is the set of *session behaviours*. The set of session behaviours is bigger than the set of session contracts because of the lack of distinction between labels and base types. It is possible to write session behaviours as

$$?\mathtt{Int}.\mathbf{1}+?\mathtt{l}_1.\mathbf{1}$$

which are image of no session type according to the given interpretation $[\![-]\!]$. Note, though, that $[\![-]\!] = \mathcal{M}$, so the range of $[\![-]\!]$ is the set of session contracts, and our Theorem 5.7 proves that $[\![-]\!]$ and their pre-order $\preceq$: Barbanera and de'Liguoro (2010, Definition 3.4) provides a complete model for the sub-typing. The completeness of $\preceq$: was only conjectured in Barbanera and de'Liguoro (2010).

Their approach is complementary to ours, in that they provide a co-inductive characterization of the pre-order $\preceq$:, which turns out to equal the intersection of their sub-server and sub-client pre-orders. In contrast, we have studied the (restricted) server and the client pre-orders independently, providing their co-inductive characterizations; we have then (1) explained why it is necessary to use the intersection of the two pre-orders to obtain a fully-abstract model; and (2) proven that the intersection of these pre-orders is a sound *and complete* model of the sub-typing.

The comparison with (Padovani 2010) is straightforward. Our definition of compliance and the definition in that paper (Definition 2.1) are different; nevertheless, we can prove that the resulting relations coincide (as long as we instantiate $\bowtie$ to the relation $\bowtie_i$). As a consequence, also our server pre-order $\sqsubseteq^{\bowtie_i}_{\mathrm{SRV}}$ coincides with the strong subcontract relation (Padovani 2010, see Definition 2.2).

6.2.2. *Fair-theories.* In Bravetti and Zavattaro (2009) and Padovani (2011), the notion of correctness requires *all* the components of a composition to be successful (i.e. be able of performing ✓) *at the same time* in order for the whole composition to be successful. This requirement implies that the compositions which contain terms as

$$\mathrm{NIL}, \quad \mu x.\, a.x$$

cannot be correct, because the contracts above do not perform ✓ at all. This phenomenon renders the viability of contracts (Padovani 2011, Definition 3.1) a non-trivial matter; on the contrary, in our theory every contract is viable wrt. MUST and $\dashv$.

The pre-orders on session contracts and session types that we have studied allow the refinement

$$a \oplus b \sqsubseteq a. \tag{14}$$

For instance, we can prove the following facts

$$\begin{array}{lll} \mu x.(\ !\mathtt{espresso}.x \oplus !\mathtt{moka}.\mathbf{1}\ ) & \sqsubseteq^{\mathcal{SC}} & \mu x.\ !\mathtt{espresso}.x \\ \mu X.\ \oplus\langle\, \mathtt{livelock} : X,\ \mathtt{stop} : \mathrm{END}\,\rangle & \preccurlyeq_{\mathrm{ST}} & \mu X.\ \oplus\langle\, \mathtt{livelock} : X\,\rangle. \end{array}$$

In Padovani (2011), it is pointed out that the refinements shown above are not sound with respect to the fair-testing (Rensink and Vogler 2007); this will let us prove that the refinements proposed in Padovani (2011) and Bravetti and Zavattaro (2009) are not contained in our relations $\sqsubseteq_{\mathrm{SRV}}$ and $\sqsubseteq^{\mathcal{SC}}$.

Now we give the detailed comparisons with the mentioned papers. The language used in Padovani (2011) is similar to our session contracts, the differences being that actions are decorated with a role tag p, q, ... ; and there is a special session type FAIL. Also, sessions are *multiparty*, that is they are general compositions of session types (tagged with a role), for instance

$$\mathtt{p}_1 : T_1 \parallel \mathtt{p}_2 : T_1 \parallel \ldots \parallel \mathtt{p}_k : T_k.$$

The notion of correct session type composition is given in Padovani (2011, Definition 2.1), and it is used to define a set-theoretical sub-typing relation on session types (Padovani 2011, Definition 2.2), which is denoted $\preccurlyeq$. We can prove

$$\mu x.(\ \mathtt{p!livelock}.x \oplus \mathtt{p!stop}.\mathbf{1}\ ) \not\preccurlyeq \mu x.\mathtt{p!livelock}.x$$

because the term $\mu x.\mathtt{p!livelock}.x$ cannot reach a successful state at all. This means that (14) is not sound for $\preccurlyeq$.

Also the following facts are true:

$$\begin{array}{rcl} \mu x.\,\texttt{p!livelock}.x & \leqslant & \mu x.\,\texttt{p!stop}.x \\ \mu x.\,\texttt{p!livelock}.x & \not\sqsubseteq^{\mathcal{SC}} & \mu x.\,\texttt{p!stop}.x \end{array}$$

The first fact is true because no composition containing the session type $\mu x.\,\texttt{p!livelock}.x$ can be correct, as this term does not perform ✓ at all; whereas the second fact follows from the definition of $\mathcal{M}$ and Definition 2.10. It thus follows that

$$\leqslant \not\subseteq \sqsubseteq^{\mathcal{SC}}, \quad \sqsubseteq^{\mathcal{SC}} \not\subseteq \leqslant .$$

Similar to Padovani (2011), in Bravetti and Zavattaro (2009) general compositions of contracts are allowed

$$[C_1] \parallel [C_2] \parallel \cdots \parallel [C_n]$$

and for a composition to be successful *all* its components have to be successful (i.e. be able of performing ✓) *at the same time*.

As for the contract language, the main difference with our framework is that the theory of Bravetti and Zavattaro (2009) involves output persistent contracts (Bravetti and Zavattaro 2009, see Definition 4); for instance the term $!a.\mathbf{1} + b.\mathbf{1}$ is a contract in our theory, that is ruled out in Bravetti and Zavattaro (2009), as it is not output persistent.

Definition 12 of that paper introduces the subcontract relations $\preceq_O$ on output persistent contracts, where the parameter O is a the set of *output* actions that the compositions used as tests can show. The comparison between our server pre-orders and the pre-orders $\preceq_O$ is complicated by two aspects,

— if $C' \preceq_O C$ then it is safe to use C' in place of C; in view of this, we will compare our pre-orders with *the inverse* of the pre-orders $\preceq_O$;
— *a priori*, it is not clear how to choose the parameter O. To solve this complication we treat $\preceq_O$ as a function of O, and briefly discuss its monotonicity.

The function $\preceq_O$ is not monotonically increasing, as $\emptyset \subseteq \{\overline{a}\}$, while

$$\begin{array}{rcl} a.\textsc{nil} & \preceq_\emptyset & a.\mathbf{1} \\ a.\textsc{nil} & \not\preceq_{\{\overline{a}\}} & a.\mathbf{1}. \end{array}$$

On the other hand $\preceq_O$ is monotonically *decreasing* in O.

Proposition. If $O \subseteq O'$ then $\preceq_{O'} \subseteq \preceq_O$.

This proposition gives us two criteria to reason on all the pre-orders $\preceq_O$:

— for every O the pairs in $\preceq_{\mathcal{N}}$ are in $\preceq_O$, where $\mathcal{N} = \{\, \overline{a} \mid a \in Act\,\}$;
— for every O the pairs *not* in $\preceq_\emptyset$ are *not* in $\preceq_O$.

As for the restriction on output persistent contracts, in the oncoming discussion we will use only contracts that enjoy that property; thus our arguments are sound.

We have the following facts

$$\begin{array}{rcl} \mathbf{1} + a.\mathbf{1} & \sqsubseteq_{\textsc{srv}} & a.\mathbf{1} \\ \mathbf{1} + a.\mathbf{1} & \not\preceq^{-1}_\emptyset & a.\mathbf{1} \end{array}$$

where the test used to prove the second fact is **1**. Also the ensuing facts are true,

$$a.\text{NIL} \preceq_{\mathcal{N}}^{-1} \text{NIL}$$
$$a.\text{NIL} \not\sqsubseteq_{\text{SRV}} \text{NIL}.$$

The first fact holds because no system containing $a.\text{NIL}$ can be correct. The test that we use to prove the second fact is $\overline{a}.\mathbf{1}$. Thus far we have proven

$$\sqsubseteq_{\text{SRV}} \not\subseteq \preceq_{\varnothing}, \quad \preceq_{\mathcal{N}} \not\subseteq \sqsubseteq_{\text{SRV}} .$$

Similarly to what done for the relation $\leqslant$ of (Padovani 2011), we can prove also that

$$\mu x.\,(\,\tau.!\mathtt{livelock}.x + \tau.!\mathtt{stop}.\mathbf{1}\,) \not\preceq_{\varnothing}^{-1} \mu x.\,!\mathtt{livelock}.x$$

Thus our relation $\sqsubseteq^{\mathcal{SC}}$ is coarser than $\preceq_{\varnothing}$. As $\preceq_{\mathcal{N}}$ relates also terms more general than session contracts, we have the following facts

$$\sqsubseteq^{\mathcal{SC}} \not\subseteq \preceq_{\varnothing}, \quad \preceq_{\mathcal{N}} \not\subseteq \sqsubseteq^{\mathcal{SC}} .$$

6.3. *Future work*

6.3.1. *Models for session types and sub-typing*. Session types have been originally used to type dialects of the pi-calculus (Honda *et al.* 1998). Recently session types have been proposed also to type other formalisms, for instance orchestration charts (Fantechi and Najm 2008). In that paper session types have behaviours described by typecharts, which are essentially LTS's (see Section 4.1 there), and the sub-typing $\preceq$, is defined in the following simulation-like manner:

$T_1 \preceq T_2$ if and only if the following conditions are true,

— if $T_2 \xrightarrow{?m} T_2'$ implies that there exists a T_1' such that $T_1 \xrightarrow{?m} T_1'$ and $T_1' \preceq T_2'$
— if $T_1 \xrightarrow{!m} T_1'$ implies that there exists a T_2' such that $T_2 \xrightarrow{!m} T_2'$ and $T_1' \preceq T_2'$.

In view of Example 5.9, it should be clear that the relation $\sqsubseteq^{\mathcal{SC}}$ does not model $\preceq$. We leave as an open problem the use of $\sqsubseteq^{\mathcal{SC}}$ to model the inverse of $\preceq$; the complication being that the definition above does not account for the τ actions.

6.3.2. *Higher-order language*. The restriction to first-order session types is a severe limitation on our results, and we intend to extend them to the full language of session types in Gay and Hole (2005). To this end it will be necessary to use a higher-order version of the language that we have used in the paper; that fact that we have used the parameter $\bowtie$ to express the notion of co-action will be of some help here.

6.3.3. *Refinements for clients*. By and large, the results in the literature on contracts for web-services pertain to refinements for either the server side of binary connections, or peers of multi-party connections. To the best of our knowledge, the first paper that avails of a refinement for clients is Barbanera and de'Liguoro (2010). Assuming the obvious definition of the refinements for the clients due to the must testing and to the compliance relation, denoted respectively $\sqsubseteq^{\text{CLT}}_{\text{MUST}}$ and $\sqsubseteq_{\text{CLT}}$, we can show that in our framework these pre-orders differ from the respective refinement for servers,

$$\begin{array}{ccccc} \sqsubseteq^{\text{CLT}}_{\text{MUST}} & \not\subseteq & \sqsubseteq_{\text{MUST}} & \not\subseteq & \sqsubseteq^{\text{CLT}}_{\text{MUST}} \\ \sqsubseteq_{\text{CLT}} & \not\subseteq & \sqsubseteq_{\text{SRV}} & \not\subseteq & \sqsubseteq_{\text{CLT}} \end{array}$$

These facts call for an investigation of the client refinements $\sqsubseteq^{\text{CLT}}_{\text{MUST}}$ and $\sqsubseteq_{\text{CLT}}$, which we leave for future work.

6.3.4. *Decidability refinements.* In Corollary 5.8, we have briefly discussed the decidability of the session pre-order. We leave as open problem the investigation of the decidability of the other refinements we have discussed.

Acknowledgements

The authors wish to acknowledge the anonymous reviewers for the criticisms and the suggestions on the previous drafts of this paper.

References

Alonso, G., Casati, F., Kuno, H. A. and Machiraju, V. (2004) *Web Services - Concepts, Architectures and Applications*, Data-Centric Systems and Applications, Springer.

Barbanera, F. and de'Liguoro, U. (2010) Two notions of sub-behaviour for session-based client/server systems In: Kutsia, T., Schreiner, W. and Fernández, M. (eds.) *PPDP*, ACM 155–164.

Bernardi, G., Bugliesi, M., Macedonio, D. and Rossi, S. (2008) *A theory of adaptable contract-based service composition.* In: Negru, V., Jebelean, T., Petcu, D. and Zaharie, D. (eds.) *SYNASC*, IEEE Computer Society 327–334.

Bernardo, M., Padovani, L. and Zavattaro, G. (eds.) (2009) In: Formal Methods for Web Services, 9th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2009, Bertinoro, Italy, Advanced Lectures. *Springer Lecture Notes in Computer Science* **5569 (June 1–6, 2009)**.

Bravetti, M. and Zavattaro, G. (2009) Contract-based discovery and composition of web services In: Bernardo *et al.* (2009) 261–295.

Caires, L. and Pfenning, F. (2010) Session types as intuitionistic linear propositions In: Gastin, P. and Laroussinie, F. (eds.) Proceedings of the CONCUR. *Springer Lecture Notes in Computer Science* **6269** 222–236.

Carpineti, S., Castagna, G., Laneve, C. and Padovani, L. (2006) A formal account of contracts for web services In: Bravetti, M., Núñez, M. and Zavattaro, G. (eds.) WS-FM. *Springer Lecture Notes in Computer Science* **4184** 148–162.

Castagna, G., Gesbert, N. and Padovani, L. (2009) A theory of contracts for web services. *ACM Transactions on Programming Languages and Systems* **31** (5) 1–61. (Supersedes the article in POPL '08.)

Eshuis, R. and Fokkinga, M. M. (2002) Comparing refinements for failure and bisimulation semantics. *Fundamenta Informaticae* **52** (4) 297–321.

Fantechi, A. and Najm, E. (2008) Session types for orchestration charts. In: Lea, D. and Zavattaro, G. (eds.) COORDINATION. *Springer Lecture Notes in Computer Science* **5052** 117–134.

Gay, S., Vasconcelos, V. and Ravara, A. (2003) Session types for inter-process communication, *Technical Report TR-2003-133*, Department of Computing Science, University of Glasgow.

Gay, S. J. and Hole, M. (2005) Subtyping for session types in the pi calculus. *Acta Informatica* **42** (2-3) 191–225.

Honda, K., Vasconcelos, V. T. and Kubo, M. (1998) Language primitives and type discipline for structured communication-based programming. In: Hankin, C. (ed.) ESOP. *Springer Lecture Notes in Computer Science* **1381** 122–138.

Laneve, C. and Padovani, L. (2007) The must preorder revisited. In: Proceedings of the 18th international conference on Concurrency Theory, Springer-Verlag, 212–225.

Laneve, C. and Padovani, L. (2008) The pairing of contracts and session types. In: Degano, P., Nicola, R. D. and Meseguer, J. (eds.) Concurrency, Graphs and Models. *Springer Lecture Notes in Computer Science* **5065** 681–700.

McNeile, A. (2010) A framework for the semantics of behavioral contracts In: Proceedings of the Second International Workshop on Behaviour Modelling: Foundation and Applications, BM-FA '10, ACM, New York, NY, USA 3:1–3:5.

Meyer, B. (1997) *Object-Oriented Software Construction*, 2nd edition, Prentice-Hall.

Milner, R. (1999) *Communicating and Mobile Systems - the Pi-calculus*, Cambridge University Press.

Nicola, R. D. and Hennessy, M. (1984) Testing equivalences for processes. *Theoretical Computer Science* **34** 83–133.

Nicola, R. D. and Hennessy, M. (1987) CCS without τ's In: TAPSOFT, Vol.1. *Springer Lecture Notes in Computer Science* **249** 138–152.

OASIS Standard (2011) Universal Description, Discovery, and Integration. Available at http://uddi.xml.org/

Padovani, L. (2010a) Contract-based discovery of web services modulo simple orchestrators. *Theoretical Computer Science* **411** (37) 3328–3347.

Padovani, L. (2011b) Fair subtyping for multi-party session types In: Proceedings of the 13th Conference on Coordination Models and Languages. *Springer Lecture Notes in Computer Science* **6721** 127–141.

Pierce, B. C. and Sangiorgi, D. (1996) Typing and subtyping for mobile processes. *Mathematical Structures in Computer Science* **6** (5) 409–453.

Rensink, A. and Vogler, W. (2007) Fair testing. *Information and Computation* **205** (2) 125–198.

Rose, M. (1988) Post Office Protocol: Version 3, RFC 1081. Obsoleted by RFC 1225.