A Note On Compliance Relations And Fixed Points.[*]

Maurizio Murgia

Università degli Studi di Cagliari

`maurizio.murgia@unica.it`

We study compliance relations between behavioural contracts in a syntax independent setting based on Labelled Transition Systems. We introduce a fix-point based family of compliance relations, and show that many compliance relations appearing in literature belong to this family.

1 Introduction

Behavioural contracts are abstract descriptions of the external behaviour and interaction scheme of distributed services [17]. They often come together with some compliance relation, which intuitively relates contracts of services whose composition is correct, where the notion of correctness is specific to the application domain [8]. In a related line of research, so called testing theories are used to study observational equivalence of CCS processes through the concept of passing a test [14]. Roughly, two processes are equivalent if they pass the same sets of tests. Tests are themselves processes, and a process passes a test when its parallel composition with the test enjoys some behavioural property (e.g., must or may reach a successful state). In retrospect, the relation between a process and a passed test can be seen as a compliance relation [18]. A selection of compliance/test relations, and their relative merits and inclusions, has been surveyed in [8] in a common ground based on Labelled Transition Systems. However, there is still lack of a general unifying theory of compliance relations, which would help to improve current practices in design and implementation of distributed concurrent systems.

Contribution. This paper is a first step towards a better understanding of the mathematical foundations of compliance relations. The starting point is a simple observation, based on two well known compliance relations: progress and must compliance. Progress relates contracts whose composition never gets stuck, or terminates in a successful state. Must relates contracts whose composition always terminates in a successful state. Intuitively, there is a duality between progress, which allows infinite behaviour, and must, which is only about finite behaviour. Two standard tools for reasoning about finiteness and infiniteness are, respectively, induction and coinduction, or, equivalently, least and greatest fixed points of monotonic functionals over complete lattices. This paper introduces a family of compliance relations, dubbed fix-compliance relations, defined as the set of fixed point of a simple and natural functional. We show that progress and must are, respectively, the greatest and the least fixed point of such compliance functional. We also consider other notions of compliance. For instance, should and behavioural compliance, which allow for infinite behaviour but with some limitations, turn out to be intermediate fixed points. Some compliance relations in literature are not fix-compliance, e.g. IO-compliance and may compliance. However, it turn out that IO-compliance is a post-fixed point, while may is a pre-fixed point.

Synopsis.   We start introducing the contract model and some notation in Section 2. We then define the compliance functional and the concept of fix-compliance in Section 3. In the rest of Section 3 we present several known compliance relations, and we show how the fit the fix-compliance framework. Section 4 discusses related works and concludes. Some rappresentative proofs are relegated to Appendix A.

2   Contracts

In this section we present a model of contracts, following the lines of [8]. Contracts are formalised as states of a Labelled Transition System (LTS) where labels are partitioned into *internal*, *input*, and *output* actions. All the compliance relations defined later on in Section 3 will be formalised as binary relations between states.

Our treatment is developed within the LTS $\left(\mathbb{U}, \mathsf{A}_\tau, \left\{\xrightarrow{\ell_\tau} \mid \ell_\tau \in \mathsf{A}_\tau\right\}\right)$, where:

- $\mathbb{U}$ is the universe of *states* (ranged over by $p, q, \ldots$), also called *contracts*;

- A_τ (ranged over by $\ell_\tau, \ell'_\tau, \ldots$) is the set of *labels*, partitioned into *input actions* $?\mathsf{a}, ?\mathsf{b}, \ldots \in \mathsf{A}^?$, *output actions* $!\mathsf{a}, !\mathsf{b}, \ldots \in \mathsf{A}^!$, and the *internal action* τ;

- $\xrightarrow{\ell_\tau} \subseteq \mathbb{U} \times \mathbb{U}$ is a *transition relation*, for all ℓ_τ.

We let $\ell, \ell', \ldots$ range over $\mathsf{A} = \mathsf{A}^? \cup \mathsf{A}^!$. We postulate an involution $\mathrm{co}(\cdot)$ on A, such that $\mathrm{co}(?\mathsf{a}) = !\mathsf{a}$ and $\mathrm{co}(!\mathsf{a}) = ?\mathsf{a}$. The *reducts of p* are the states reachable from p with a finite sequence of transitions with any label, while the ℓ_τ-*reducts of p* are the states reachable from p with a finite sequence of transitions with label ℓ_τ. A *trace* is a (possibly infinite) sequence $p_0 \xrightarrow{\ell_\tau{}^{(1)}} p_1 \xrightarrow{\ell_\tau{}^{(2)}} \cdots$. A τ-trace is a trace where $\ell_\tau{}^{(i)} = \tau$, for all i (similarly for τ-reduct). We assume that there exists a unique state with no outgoing transitions. Such state is denoted by $\mathbf{0}$. Note that, since $\mathbf{0}$ is unique, if p is such that $p \xnrightarrow{\ell_\tau}$ for all ℓ_τ, then $p = \mathbf{0}$. We interpret $\mathbf{0}$ as a correctly terminated state, and we will often refer to $\mathbf{0}$ as the *success* state.

Notation 1. *We adopt the following notation:*

- $\mathcal{R}^*$ *for the reflexive and transitive closure of a relation $\mathcal{R}$*

- $p \xrightarrow{\ell_\tau}$ *when $\exists p' . p \xrightarrow{\ell_\tau} p'$. Further, we write $p \rightarrow$ when $\exists \ell_\tau . p \xrightarrow{\ell_\tau}$*

- *for a set $L \subseteq \mathsf{A}$, we define $L^? = L \cap \mathsf{A}^?$ and $L^! = L \cap \mathsf{A}^!$*

- $\Rightarrow = (\xrightarrow{\tau})^*$ *is the* weak transition relation. *We define $\xRightarrow{\ell_\tau}$ as $\Rightarrow \xrightarrow{\ell_\tau} \Rightarrow$*

- $p\!\downarrow = \left\{\ell \mid p \xrightarrow{\ell}\right\}$ *are the* barbs of p, *and $p\!\Downarrow = \left\{\ell \mid p \xRightarrow{\ell}\right\}$ are its* weak barbs

- $p\!\uparrow$ *is true when p has an infinite internal computation $p \xrightarrow{\tau} p_1 \xrightarrow{\tau} p_2 \xrightarrow{\tau} \cdots$*

The above notation for $\rightarrow$ is extended to $\Rightarrow$ as expected.

In order to define parallel composition of contracts, we require some additional structure on $\mathbb{U}$. In particular, we assume $\mathbb{U}$ to be closed under a binary operation $\parallel$. Contracts in the form $p \parallel q$ are called *compositions*, and we refer to the left component p as the *client* and the right component q as the *server*. Compositions where the client is $\mathbf{0}$ are called *successful* , and we refer to the set of all successful compositions as $\mathcal{S}$. Formally, $\mathcal{S} = \{\mathbf{0} \parallel p \mid p \in \mathbb{U}\}$. Note that $\mathbf{0}$ models success of a single participant, while the elements of $\mathcal{S}$ model success of compositions of (at least) two participants. Intuitively, $\mathcal{S}$ contains all compositions in which the client is terminated, and so in which the server has successfully satisfied the client. This asymmetric notion can be found in previous work [5].

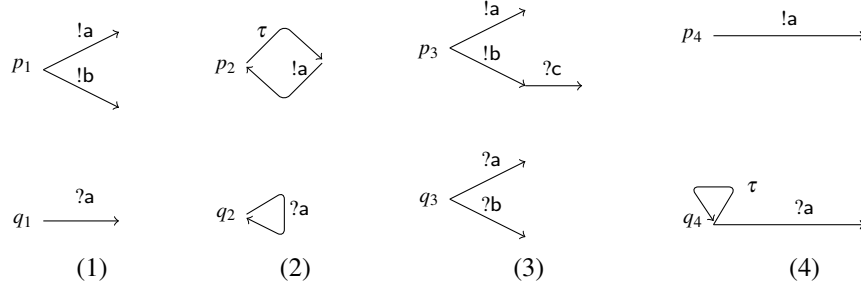The semantics of compositions formalises the standard synchronisation à la CCS [20].

Figure 1: Some pairs of contracts.

Definition 1 (Parallel composition)**.** *For all $p, q \in \mathbb{U}$, we impose $p \parallel q \in \mathbb{U}$. The transition relation of compositions contains all and only the transitions that can be derived with the following rules:*

$$\frac{p \xrightarrow{\ell_\tau} p'}{p \parallel q \xrightarrow{\ell_\tau} p' \parallel q} \qquad \frac{q \xrightarrow{\ell_\tau} q'}{p \parallel q \xrightarrow{\ell_\tau} p \parallel q'} \qquad \frac{p \xrightarrow{\ell} p' \quad q \xrightarrow{\mathrm{co}(\ell)} q'}{p \parallel q \xrightarrow{\tau} p' \parallel q'}$$

3 A fixed-point based family of compliance relations

In this section we introduce a general class of compliance relations between behaviours, based on the compliance functional $\mathcal{C}$ defined below. We then show that many compliance relations in literature, but not all, fit within this class. Compliance relation in this class have the following properties:

- contracts whose composition is successful are compliant;

- compositions of compliant contracts never get stuck before a successful state is reached;

- compliance is preserved by τ-transitions, until a successful state is reached.

Definition 2. *We define the compliance functional $\mathcal{C} : \mathbb{U}^2 \to \mathbb{U}^2$ as follows:*

$$\mathcal{C}(x) = \mathcal{S} \cup \left\{ (p, q) \,\middle|\, p \parallel q \xrightarrow{\tau} \wedge (p \parallel q \xrightarrow{\tau} p' \parallel q' \implies (p', q') \in x) \right\}$$

We say that a relation $\mathcal{R} \subseteq \mathbb{U}^2$ is:

- *a* pre-*compliance relation if $\mathcal{R}$ is a pre-fixed point of $\mathcal{C}$, that is $\mathcal{C}(\mathcal{R}) \subseteq \mathcal{R}$;*

- *a* post-*compliance relation if $\mathcal{R}$ is a post-fixed point of $\mathcal{C}$, that is $\mathcal{R} \subseteq \mathcal{C}(\mathcal{R})$;*

- *a* fix-*compliance relation if $\mathcal{R}$ is a fixed-point of $\mathcal{C}$, that is $\mathcal{R} = \mathcal{C}(\mathcal{R})$.*

We start recalling that, by the Knaster-Tarski theorem [22], every monotonic endo-function over a complete lattice has a least fixed point and a greatest fixed point (they may coincide). Furthermore, the least fixed-point coincides with the least pre-fixed point and the greatest fixed point coincides with the greatest post-fixed point. We will now on work on the complete lattice $\mathbb{U} \times \mathbb{U}$ ordered by set inclusion. It is easy to verify that $\mathcal{C}$ is monotonic with respect to $\subseteq$, that is, for all $x, y \subseteq \mathbb{U} \times \mathbb{U}$:

$$x \subseteq y \implies \mathcal{C}(x) \subseteq \mathcal{C}(y)$$

Progress compliance. We start by considering the notion of *progress*, which consists of absence of deadlocks (on the client-side, since we are considering the asymmetric relation). Formally, in Definition 3 we say that a contract p has progress with q (in symbols, $p \lhd_{pg} q$) iff, whenever a τ-reduct of $p \parallel q$ is stuck, then p has reached the success state.

Definition 3 (Progress). *We write $p \lhd_{pg} q$ iff:*

$$p \parallel q \ \Rightarrow \ p' \parallel q' \xrightarrow{\tau}\!\!\!\!\!/ \quad \text{implies} \quad p' = \mathbf{0}$$

This notion has been used e.g. in τ-less CCS [13], in session types (both untimed [5] and timed [6]), and in types for CaSPiS [2].

Example 1. *Consider the behaviours in fig. 1.*

- *We have that $p_1 \lhd_{pg} q_1$: the composition $p_1 \parallel q_1$ can only τ-reduce through a synchronisation on* a, *leading to a successful state.*

- *The composition $p_2 \parallel q_2$ can only take the p_2 τ-move, and then synchronise on* a, *going back to the starting state. Therefore, $p_2 \lhd_{pg} q_2$.*

- *The composition $p_3 \parallel q_3$ may τ-reduce through a synchronisation on* b, *leading to a state which is stuck (no τ-reductions are possible) but unsuccessful (p_3 is not terminated as she can emit a* ?c *action). Therefore, $p_3 \not\lhd_{pg} q_3$.*

- *The composition $p_4 \parallel q_4$ can loop taking the p_4 τ-move, or τ-reduce to a successful state through a synchronisation on* a. *Therefore, $p_4 \lhd_{pg} q_4$.*

It turns out that $\lhd_{pg}$ is the largest fix-compliance.

Proposition 1. $\lhd_{pg}$ *is the largest fix-compliance.*

An important consequence of Proposition 1 is that all post-compliance relations enjoy the progress property (as defined in Definition 3): indeed, if x is a post-compliance, then, by the Knaster-Tarki Theorem it follows $x \subseteq \lhd_{pg}$.

Must-testing compliance. The notion of compliance in [3] is inspired to must-testing [14]. Must testing requires a contract to reach success in *all* (sufficiently long) traces. Formally, we say that a τ-trace $r_0 \to r_1 \to \cdots$ is *maximal* if it is infinite, or if it ends in a state r_n such that $r_n \xrightarrow{\tau}\!\!\!\!\!/$. A behaviour p is must-testing compliant with q (in symbols, $p \lhd_{mst} q$) if, in all the maximal τ-traces of $p \parallel q$, the contract p reaches the $\mathbf{0}$ state.

Definition 4 (Must-testing compliance). *We write $p_0 \lhd_{mst} q_0$ iff*

$$\text{for all maximal } \tau\text{-traces } p_0 \parallel q_0 \xrightarrow{\tau} p_1 \parallel q_1 \xrightarrow{\tau} \cdots \ : \ \exists i \geq 0 \,.\, p_i = \mathbf{0}$$

Example 2. *Consider the behaviours in fig. 1.*

- $p_1 \lhd_{mst} q_1$: *the only maximal τ-trace is $p_1 \parallel q_1 \xrightarrow{\tau} \mathbf{0} \parallel \mathbf{0}$, which contains a composition whose left component is $\mathbf{0}$.*

- $p_2 \not\lhd_{mst} q_2$: *the composition $p_2 \parallel q_2$ diverges without visiting a successful state.*

- $p_3 \not\lhd_{mst} q_3$, *basically for the same reason of Example 1.*

- $p_4 \not\lhd_{mst} q_4$: *the composition $p_4 \parallel q_4$ may perpetually loop taking the p_4 τ-move, without visiting any successful state.*

Proposition 2. $\lhd_{mst}$ *is the least fix-compliance relation.*

Should-testing compliance.   We now present a notion of compliance inspired by the theory of should-testing [12, 21]. A behaviour p is *should-testing compliant* with q (in symbols, $p \lhd_{shd} q$) if, after every possible *finite* τ-trace of $p \parallel q$, there exists a subsequent (finite) τ-trace which leads p to the success state.

Definition 5 (Should-testing compliance)**.** *We write $p \lhd_{shd} q$ iff*

$$p \parallel q \; \Rightarrow \; p' \parallel q' \;\; \text{implies} \;\; \exists q'' \; . \; p' \parallel q' \Rightarrow \mathbf{0} \parallel q''$$

A notion similar to the one in Definition 5 has been used in [11] (under the name of *correct contract composition*), and in [1, 7] (where it is named *weak termination*).

Example 3. *Consider the behaviours in fig. 1.*

- *$p_1 \lhd_{shd} q_1$: the composition $p_1 \parallel q_1$ can only τ-reduce through a synchronisation on a, leading to a successful state.*

- *$p_2 \not\lhd_{shd} q_2$. As noted in Example 2, the composition $p_2 \parallel q_2$ necessarily diverges, and no successful state is reachable.*

- *$p_3 \not\lhd_{shd} q_3$, for the same reason of Examples 1 and 2.*

- *$p_4 \lhd_{shd} q_4$. The composition $p_4 \parallel q_4$ can loop taking the p_4 τ-move, but a successful state is invariantly reachable through a synchronisation on a.*

Proposition 3. *$\lhd_{shd}$ is a fix-compliance relation.*

Behavioural compliance.   Definition 6 below formalises in our setting the relation called *behavioural compliance* in [18, 19]. A contract p is compliant with q (in symbols, $p \lhd_{beh} q$), if, in every possible τ-reduct $p' \parallel q'$ of $p \parallel q$, two conditions are satisfied: if the reduct is stuck, then p' has reached success; otherwise, if q' alone can produce an infinite τ-trace, then p' must be able to reach success without further synchronisations.

Definition 6 (Behavioural compliance)**.** *We write $p \lhd_{beh} q$ iff:*

$$p \parallel q \; \Rightarrow \; p' \parallel q' \;\; \text{implies} \;\; \left(p' \parallel q' \overset{\tau}{\not\Rightarrow} \; \text{implies} \; p' = \mathbf{0} \right) \wedge \left(q'\!\uparrow \text{implies} \; p' \Rightarrow \mathbf{0} \right)$$

Example 4. *Consider the behaviours in fig. 1.*

- *$p_1 \lhd_{beh} q_1$: q_2 does not diverge, and the composition $p_1 \parallel q_1$ can only τ-reduce through a synchronisation on a, leading to a successful state.*

- *$p_2 \lhd_{beh} q_2$: as noted in Example 1, the composition $p_2 \parallel q_2$ never gets stuck, and q_2 does not diverge.*

- *$p_3 \not\lhd_{beh} q_3$, for the same reason of Examples 1 to 3.*

- *$p_4 \not\lhd_{beh} q_4$: Although the composition $p_4 \parallel q_4$ never gets stuck, q_4 may diverge and p_4 cannot terminate on her own.*

Proposition 4. *$\lhd_{beh}$ is a fix-compliance.*

I/O compliance. In [9], a contract p is considered compliant with q (in symbols, $p \lhd_{io} q$), if, in every possible τ-reduct $p' \| q'$ of $p \| q$, the weak outputs of p' are included in the weak inputs of q'; further, if p' has no weak outputs but still some weak inputs, then they include the weak outputs of q'.

Definition 7 (I/O compliance). *We write $p \lhd_{io} q$ iff $p \| q \Rightarrow p' \| q'$ implies:*

$$p' \Downarrow^! \subseteq \mathrm{co}(q' \Downarrow^?) \ \wedge \ \left((p' \Downarrow^! = \emptyset \wedge p' \Downarrow^? \neq \emptyset) \implies \emptyset \neq q' \Downarrow^! \subseteq \mathrm{co}(p' \Downarrow^?)\right)$$

Example 5. *Consider the behaviours in fig. 1.*

- $p_1 \not\lhd_{io} q_1$: $p_1 \Downarrow^! = \{!a, !b\} \not\subseteq \{!a\} = \mathrm{co}(q_1 \Downarrow^?)$.

- $p_2 \lhd_{io} q_2$: *we have that, in every τ-reduct $p'_2 \| q'_2$ of $p_2 \| q_2$, $p'_2 \Downarrow^! = \{!a\}$ and $\mathrm{co}(q'_2 \Downarrow^?) = \{!a\}$. Therefore both conjuncts of Definition 7 holds.*

- $p_3 \not\lhd_{io} q_3$: *after a synchronisation on* b, *a state $p'_3 \| q'_3$ is reached. However, $p'_3 \Downarrow^! = \emptyset$ and $p'_3 \Downarrow^? \neq \emptyset$, but $q'_3 \Downarrow^! = \emptyset$. Therefore, the second conjunct of Definition 7 does not hold.*

- $p_4 \lhd_{io} q_4$: *The only reachable states are $p_4 \| q_4$ and $\mathbf{0} \| \mathbf{0}$. As $p'_2 \Downarrow^! = \{!a\}$ and $\mathrm{co}(q'_2 \Downarrow^?) = \{!a\}$, $p_4 \| q_4$ satisfies the condition of Definition 7. For $\mathbf{0} \| \mathbf{0}$, it does hold as well: $\mathbf{0} \Downarrow^! = \emptyset = \mathrm{co}(\mathbf{0} \Downarrow^?)$.*

It turns out that $\lhd_{io}$ is a post-compliance but not a pre-compliance (and hence not a fix-compliance). To see why it is not a pre-compliance, consider p_1 and q_1 from fig. 1. As noted in Example 5, $p_1 \not\lhd_{io} q_1$. However, $p_1 \| q_1 \xrightarrow{\tau}$ and its unique τ-reduct is successful and hence composed by compliant behaviours. Therefore, $(p_1, q_1) \in \mathcal{C}(\lhd_{io})$.

Proposition 5. *$\lhd_{io}$ is a post-compliance relation.*

May-testing compliance. In Definition 8, a contract p is said to be *may-testing compliant* with q (in symbols, $p \lhd_{may} q$) if there exists a finite τ-trace of $p \| q$ which leads p to the success state.

Definition 8 (May-testing compliance). *We write $p \lhd_{may} q$ iff*

$$\exists q' \ . \ p \| q \Rightarrow \mathbf{0} \| q'$$

Example 6. *Consider the behaviours in fig. 1.*

- $p_1 \lhd_{may} q_1$: *$p_1 \| q_1$ can reach a successful state after a synchronisation on* a.

- $p_2 \not\lhd_{may} q_2$: *as noted in Example 3, the composition $p_2 \| q_2$ never reach any successful state.*

- $p_3 \lhd_{may} q_3$: *$p_3 \| q_3$ can reach a successful state after a synchronisation on* a.

- $p_4 \lhd_{may} q_4$: *$p_4 \| q_4$ can reach a successful state after a synchronisation on* a.

In a sense, may-testing compliance assumes a *cooperative* scenario: participants pre-agree on their internal choices, and the scheduler to only permit the synchronisations leading to success, seen here as a common goal.

It turns out that $\lhd_{may}$ is a pre-compliance relation but not a post-compliance relation (and hence not a fix-compliance). To see why it is not a post-compliance, consider p_3 and q_3 of fig. 1. As noted in Example 6, $p_3 \lhd_{may} q_3$. However, $(p_3, q_3) \notin \mathcal{C}(\lhd_{may})$: through a synchronisation on b, $p_3 \| q_3$ can reduce to a composition which is not successful nor composed by may-compliant behaviours.

Proposition 6. *$\lhd_{may}$ is a pre-compliance relation.*

4   Conclusions and related work.

Behavioural contracts and compliance relations have been studied in several works and contexts, e.g. service-oriented computing [2, 3, 11, 13, 18, 19, 1] and session types [5, 6, 9]. Testing preorders have been studied in [14, 18, 21]. The definition of testing compliance in this work is slightly different from the classical ones [14, 18, 21]: there, the successful states are those that can emit the special label *e*. Following [8], we consider **0** as the success state. This makes our treatment simple and uniform. The work [8] presents a taxonomy of compliance relations in a general setting based on LTS similar to the one used in this paper, but they also study certain subclasses of the model, which correspond to known contract models or process algebras: session types [16], τ-less CCS [15], contract automata [10] and interface automata [4]. Our work, instead, studies only on the full model, focusing on the mathematical foundations, and revealing the important role of the compliance functional $\mathcal{C}$. Among the compliance relations surveyed in [8], only IA-compliance (inspired to Interface Automata compatibility) does not seem to be related to $\mathcal{C}$ in any way. This seems due to the fact that Interface Automata, being naturally suited for modelling systems composed of many components, do not fit well our binary setting.

We have introduced a family of compliance relations, showing how different treatments to divergences in distributed systems correspond to different fixed-point of a general functional. In particular:

- Must compliance, which disallows any form of divergence, is the least fixed-point of $\mathcal{C}$.

- Should compliance relates contracts whose composition may diverge, but only if a successful terminated state is always reachable. In a sense, should assumes fairness (but not full cooperation) of participants and the scheduler to reach a success state. This form of fairness is captured as an intermediate fixed-point of $\mathcal{C}$.

- Behavioural compliance relates contracts whose composition may diverge, but forbids situations in which divergence of the server disallows the client to successfully terminate. In this case the server is considered adversarial. Also this compliance is an intermediate fixed-point of $\mathcal{C}$.

- Progress compliance allows any form of divergence, and is indeed the greatest fixed point of $\mathcal{C}$.

We have shown two examples of compliance relations appearing in literature that are not fixed-point of $\mathcal{C}$, but turn out to be pre- or post-fixed point of it. Post-compliance relations, like IO-compliance, still guarantee the good behavioural properties reported in Section 3, namely stuck-freedom and preservation of compliance by τ-reduction, but somehow relate fewer contracts than expected. In the specific case of IO-compliance, this is caused by the asymmetric treatment of outputs and inputs. The case of may compliance, is quite enigmatic: may compliance, being "cooperative" in nature [8], is out of the scope of fix-compliance relations, which are biased towards the non-cooperative scenario, but still may is a pre-compliance, and so fits somehow in our setting. It is still unclear to us whether this can lead to useful consequences, or it holds just by coincidence.

A possible future direction is the study of cooperative compliance relations through fixed-points. For instance, we would expect may compliance to be the least fixed-point of some suitable functional. We expect the greatest fixed-point of such functional to be a kind of cooperative progress, relating contracts whose composition produces at least one execution which is infinite or terminates in a successful state. An interesting but challenging future direction is characterising the subcontract preorders [13] induced by fix-compliance relations.

References

[1] Wil M. P. van der Aalst, Niels Lohmann, Peter Massuthe, Christian Stahl & Karsten Wolf (2010): *Multi-party Contracts: Agreeing and Implementing Interorganizational Processes*. *Comput. J.* 53(1), pp. 90–106, doi:10.1093/comjnl/bxn064.

[2] Lucia Acciai & Michele Boreale (2008): *A Type System for Client Progress in a Service-Oriented Calculus*. In: *Concurrency, Graphs and Models, Essays Dedicated to Ugo Montanari on the Occasion of His 65th Birthday*, pp. 642–658, doi:10.1007/978-3-540-68679-8_40.

[3] Lucia Acciai, Michele Boreale & Gianluigi Zavattaro (2010): *Behavioural Contracts with Request-Response Operations*. In: *Proc. COORDINATION*, pp. 16–30, doi:10.1007/978-3-642-13414-2_2.

[4] Luca de Alfaro & Thomas A. Henzinger (2001): *Interface automata*. In: *Proc. ACM SIGSOFT*, pp. 109–120, doi:10.1145/503209.503226.

[5] Franco Barbanera & Ugo de'Liguoro (2015): *Sub-behaviour relations for session-based client/server systems*. *Mathematical Structures in Computer Science* 25(6), pp. 1339–1381, doi:10.1017/S096012951400005X.

[6] Massimo Bartoletti, Tiziana Cimoli & Maurizio Murgia (2017): *Timed Session Types*. *Logical Methods in Computer Science* 13(4), doi:10.23638/LMCS-13(4:25)2017.

[7] Massimo Bartoletti, Tiziana Cimoli & G. Michele Pinna (2015): *Lending Petri nets*. *Sci. Comput. Program.* 112, pp. 75–101, doi:10.1016/j.scico.2015.05.006.

[8] Massimo Bartoletti, Tiziana Cimoli & Roberto Zunino (2015): *Compliance in Behavioural Contracts: A Brief Survey*. In: *Programming Languages with Applications to Biology and Security*, *Lecture Notes in Computer Science* 9465, Springer, pp. 103–121, doi:10.1007/978-3-319-25527-9_9.

[9] Massimo Bartoletti, Alceste Scalas & Roberto Zunino (2014): *A Semantic Deconstruction of Session Types*. In: *Proc. CONCUR*, pp. 402–418, doi:10.1007/978-3-662-44584-6_28.

[10] Davide Basile, Pierpaolo Degano & Gian Luigi Ferrari (2014): *Automata for Analysing Service Contracts*. In: *Proc. TGC*, pp. 34–50, doi:10.1007/978-3-662-45917-1_3.

[11] Mario Bravetti & Gianluigi Zavattaro (2007): *Contract Based Multi-party Service Composition*. In: *Proc. FSEN*, *LNCS* 4767, pp. 207–222, doi:10.1007/978-3-540-75698-9_14.

[12] Ed Brinksma, Arend Rensink & Walter Vogler (1995): *Fair Testing*. In: *Proc. CONCUR*, pp. 313–327, doi:10.1007/3-540-60218-6_23.

[13] Giuseppe Castagna, Nils Gesbert & Luca Padovani (2009): *A theory of contracts for Web services*. *ACM TOPLAS* 31(5), pp. 19:1–19:61, doi:10.1145/1538917.1538920.

[14] Rocco De Nicola & Matthew Hennessy (1984): *Testing Equivalences for Processes*. *Theor. Comput. Sci.* 34, pp. 83–133, doi:10.1016/0304-3975(84)90113-0.

[15] Rocco De Nicola & Matthew Hennessy (1987): *CCS without tau's*. In: *Proc. TAPSOFT*, pp. 138–152, doi:10.1007/3-540-17660-8_53.

[16] Kohei Honda, Vasco T. Vasconcelos & Makoto Kubo (1998): *Language Primitives and Type Disciplines for Structured Communication-based Programming*. In: *Proc. ESOP*, *LNCS* 1381, pp. 122–138, doi:10.1007/BFb0053567.

[17] Hans Hüttel, Ivan Lanese, Vasco T. Vasconcelos, Luís Caires, Marco Carbone, Pierre-Malo Deniélou, Dimitris Mostrous, Luca Padovani, António Ravara, Emilio Tuosto, Hugo Torres Vieira & Gianluigi Zavattaro (2016): *Foundations of Session Types and Behavioural Contracts*. *ACM Comput. Surv.* 49(1), pp. 3:1–3:36, doi:10.1145/2873052.

[18] Cosimo Laneve & Luca Padovani (2007): *The* Must *Preorder Revisited*. In: *Proc. CONCUR*, pp. 212–225, doi:10.1007/978-3-540-74407-8_15.

[19] Cosimo Laneve & Luca Padovani (2015): *An algebraic theory for Web service contracts*. *Formal Aspects of Computing*, pp. 1–28, doi:10.1007/s00165-015-0334-2.

[20] Robin Milner (1989): *Communication and concurrency*. Prentice-Hall, Inc.

[21] Arend Rensink & Walter Vogler (2007): *Fair testing*. Information and Computation 205(2), pp. 125–198, doi:10.1016/j.ic.2006.06.002.

[22] Alfred Tarski: *A lattice-theoretical fixpoint theorem and its applications*. Pacific J. Math. (2), pp. 285–309, doi:10.2140/pjm.1955.5.285. Available at http://projecteuclid.org/euclid.pjm/1103044538.

A Proofs

Proof of Proposition 2 on page 41

Proof. According to the Knaster-Tarski theorem, it suffice to show that $\lhd_{mst}$ is the least pre-fixed point of $\mathcal{C}$. In turn, this can proved by showing that must is a pre-fixed point of $\mathcal{C}$, and that any other pre-fixed point of $\mathcal{C}$ is larger than $\lhd_{mst}$.

- For the first part, we have to show that $\mathcal{C}(\lhd_{mst}) \subseteq \lhd_{mst}$. So, let $(p,q) \in \mathcal{C}(\lhd_{mst})$. If $(p,q) \in \mathcal{S}$, it must be $p = \mathbf{0}$, and so for every maximal τ-trace $p_0 \| q_0 \xrightarrow{\tau} p_1 \| q_1 \xrightarrow{\tau} \ldots$, with $p_0 = p$ and $q_0 = q$, there is i such that $p_i = \mathbf{0}$: just take $i = 0$. If $(p,q) \notin \mathcal{S}$, by definition of $\mathcal{C}$, we have that $p \| q \xrightarrow{\tau}$, and, for all $p' \| q'$ such that $p \| q \xrightarrow{\tau} p' \| q'$, it holds that $p' \lhd_{mst} q'$ (1). Let $p \| q \xrightarrow{\tau} p_1 \| q_1 \xrightarrow{\tau} \ldots$ be a maximal trace. Note that, by (1), it holds that $p_1 \lhd_{mst} q_1$. Therefore, the maximal trace $p_1 \| q_1 \xrightarrow{\tau} \ldots$ eventually reaches a state whose left contract is $\mathbf{0}$, and thus also the trace $p \| q \xrightarrow{\tau} p_1 \| q_1 \xrightarrow{\tau} \ldots$, as required.

- For the second part, let X be a pre-fixed point of $\mathcal{C}$, i.e. $\mathcal{C}(X) \subseteq X$. We have to show $\lhd_{mst} \subseteq X$. So, let $p \lhd_{mst} q$. If $p = \mathbf{0}$, it must be $(p,q) \in \mathcal{S}$, and so, by definition of $\mathcal{C}$, it follows $(p,q) \in \mathcal{C}(X)$. Since $\mathcal{C}(X) \subseteq X$ by assumption, we have that $(p,q) \in X$, as required. If $p \neq \mathbf{0}$, first note that it must be $p \| q \xrightarrow{\tau}$. Indeed, if this is not the case, the only maximal τ-trace starting from $p \| q$ would be $p \| q$ (seen as a singleton trace), which of course does not reach a success state as p is not $\mathbf{0}$ by assumption. Now suppose, by contradiction, $(p,q) \notin X$. Note that, since $\mathcal{C}(X) \subseteq X$, it must be $(p,q) \notin \mathcal{C}(X)$. Therefore, by definition of $\mathcal{C}$, there must be $(p_1,q_1) \notin X$ such that $p \| q \xrightarrow{\tau} p_1 \| q_1$. As before, $(p_1,q_1) \notin \mathcal{C}(X)$. Iterating the argument again and again, we can construct an infinite maximal τ-trace $p_0 \| q_0 \xrightarrow{\tau} p_1 \| q_1 \xrightarrow{\tau} \ldots$ (with $p_0 = p$ and $q_0 = q$) such that $(p_i,q_i) \notin \mathcal{C}(X)$ for all i. Then, since $\mathcal{S} \subseteq \mathcal{C}(X)$ by definition of $\mathcal{C}$, it must be $(p_i,q_i) \notin \mathcal{S}$ for all i. Then, there is maximal τ-trace starting from $p \| q$ that do not reach a success state, contradicting the hypothesis that $p \lhd_{mst} q$.

$\square$

Proof of Proposition 5 on page 43

Proof. We have to show that $\lhd_{io} \subseteq \mathcal{C}(\lhd_{io})$. So, let $p \lhd_{io} q$. The case where $p = \mathbf{0}$ is immediate, as $(p,q) \in \mathcal{S} \subseteq \mathcal{C}(\lhd_{io})$. For the remaining case $p \neq \mathbf{0}$, we have to show that $p \| q \xrightarrow{\tau}$ and that for all p',q' such that $p \| q \xrightarrow{\tau} p' \| q'$ it holds that $p' \lhd_{io} q'$. Since $p \neq \mathbf{0}$, it must be $p \xrightarrow{\ell_\tau}$ for some ℓ_τ. So, if $\ell_\tau = \tau$, the thesis follows by the first rule of parallel composition. If $\ell_\tau \in \mathsf{A}^!$, then $\ell_\tau \in p\Downarrow^!$ and so, by the first conjunct in the definition of $\lhd_{io}$, we have that $p\Downarrow^! \subseteq \mathrm{co}(q\Downarrow^?)$. Therefore, $q \xRightarrow{\mathrm{co}(\ell_\tau)}$. If the first transition of such reduction is τ, the thesis follows by an application of the second rule of parallel composition. If the first transition is ℓ_τ, the thesis follows by an application of the third rule. In the remaining case $\ell_\tau \in \mathsf{A}^?$, we have that $\ell_\tau \in p\Downarrow^? \neq \emptyset$. So, if $p\Downarrow^! \neq \emptyset$, we can conclude similarly to the previous case. If $p\Downarrow^! = \emptyset$, by the second conjunct in the definition of $\lhd_{io}$, we can conclude that $\mathrm{co}(\ell_\tau) \in p\Downarrow^?$. We can then conclude similarly to the previous case. It remain to show that for all p',q' such that $p \| q \xrightarrow{\tau} p' \| q'$ it holds $p' \lhd_{io} q'$. But this follows immediately by the definition of $\lhd_{io}$, because $p' \| q'$ is a τ-reduct of $p \| q$. $\square$