A note on two notions of compliance

Massimo Bartoletti
Dipartimento di Matematica e Informatica
University of Cagliari, Italy
bart@unica.it

Tiziana Cimoli
Dipartimento di Matematica e Informatica
University of Cagliari, Italy
t.cimoli@unica.it

G. Michele Pinna
Dipartimento di Matematica e Informatica
University of Cagliari, Italy
gmpinna@unica.it

We establish a relation between two models of contracts: binary session types, and a model based on event structures and game-theoretic notions. In particular, we show that compliance in session types corresponds to the existence of certain winning strategies in game-based contracts.

1 Introduction

Several recent papers have been devoted to the study of *contracts* as a way to formally specify abstractions of the behaviour of software systems. A common aspect that gathers together some of these studies is a notion of *compliance*. This is a relation between systems which want to interact. Before starting the interaction, contracts are statically checked for compliance: when enjoyed, it guarantees that systems respecting their contracts will interact correctly. Since distributed applications are often constructed by dynamically discovering and composing services published by different (possibly distrusting) organizations, compliance becomes relevant to protect those services from each other's misbehaviour. Indeed, the larger an application is, the greater is the probability that some of its components deviates from the expected behaviour (either because of unintentional bugs, or maliciousness).

To obtain protection, compliance can be modelled in many different ways. Typically, it is formalised as a fairness property, which ensures progress (possibly, until reaching a success state [8, 2]), or which ensures the possibility of always reaching success from any state [7, 1]. Weaker variants of compliance allow services to discard some messages [3], or involve orchestrators which can suitably rearrange them [11].

While all the above approaches express contracts as processes of some process algebra, in [4] contracts are modelled as multi-player concurrent games, whose moves are transitions in an event structure [12], and where compliance is defined as the existence of winning strategies in these games. By abstracting away from the concrete details of process calculi, this model may be used as a unifying framework for reasoning about contracts, in the same spirit that event structures are used as an underlying semantics for a variety of concrete models of concurrency.

As a first step towards unifying different views of contracts, in this paper we interpret binary session types [9] as game-based contracts, by providing them with an event structure semantics (Definition 11). Our main technical contribution is that compliance in the former model corresponds to the existence of a certain kind of winning strategies in the latter (Theorem 16). The constructions used to obtain this result suggest that also other notions of compliance (*e.g.*, *I/O compliance* [5], Padovani's *weak compliance* [11] and Barbanera & de' Liguoro's `skp`-*compliance* [3]) might be expressed game-theoretically by suitably adjusting the event structure semantics, and by restricting the class of admissible winning strategies.

I. Lanese, A. Lluch Lafuente, A. Sokolova, H. T. Vieira (Eds.):
7th Interaction and Concurrency Experience (ICE 2014)
EPTCS 166, 2014, pp. 86–93, doi:10.4204/EPTCS.166.9

$$\overline{a}.P \oplus Q \to \overline{a}.P \qquad \overline{a}.P \xrightarrow{\overline{a}} P \qquad \frac{P \to P'}{P \parallel Q \to P' \parallel Q} \qquad \frac{P \xrightarrow{\overline{a}} P' \quad Q \xrightarrow{a} Q'}{P \parallel Q \to P' \parallel Q'}$$

$$a.P + Q \xrightarrow{a} P \qquad \mathit{rec}\, x.\, P \to P\{^{\mathit{rec}\, x.\, P}/_{x}\}$$

Figure 1: Operational semantics of session types (symmetric rules omitted).

2 Session types

Let $\mathbf{A}$ be a set of *actions*, ranged over by $a, b, \ldots$, and let $\overline{\mathbf{A}} = \{\overline{a} \mid a \in \mathbf{A}\}$ be such that $\mathbf{A} \cap \overline{\mathbf{A}} = \emptyset$. We let $\alpha, \beta, \ldots$ range over $\mathbf{A} \cup \overline{\mathbf{A}}$. In Definition 1 we introduce the syntax of *binary* session types, following the notation used in [2].

Definition 1 (**Session type**). Session types *are defined as follows:*

$$P, Q \;::=\; \mathbf{1} \;\mid\; \textstyle\bigoplus_{i\in I} \overline{a}_i.P_i \;\mid\; \sum_{i\in I} a_i.P_i \;\mid\; \mathit{rec}\, x.\, P \;\mid\; x$$

where (i) *the index set I is finite and non-empty,* (ii) *the actions in internal/external choices are pairwise distinct, and* (iii) *recursion is guarded.*

Session types are processes of a process algebra featuring $\mathbf{1}$ (success), internal choice $\bigoplus_{i\in I} \overline{a}_i.P_i$, external choice $\sum_{i\in I} a_i.P_i$, and guarded recursion. If $Q = \bigoplus_{i\in I} \overline{a}_i.P_i$ and $0 \notin I$, we write $\overline{a_0}.P_0 \oplus Q$ for $\bigoplus_{i\in I\cup\{0\}} \overline{a}_i.P_i$ (same for external choice).

The semantics of session types is defined in Figure 1. The intuition is that a session type models the intended behaviour of *one* of the two participants involved in a session, while the behaviour of two interacting participants is modelled by the composition of two session types, denoted $P \parallel Q$. An internal choice must first commit to one of the branches $\overline{a}.P$, before advertising $\overline{a}$. An external choice can always advertise each of its actions. There, participants can run asynchronously only when committing to a branch or unfolding recursion. Synchronisation requires that a participant has committed to a branch $\overline{a}$ in an internal choice, and the other offers a in an external choice.

Following [10, 8, 2] we define a notion of compliance between session types. The intuition is that if a client contract P is compliant with a server contract Q then, whenever a computation of $P \parallel Q$ becomes stuck, the client has reached the success state.

Definition 2 (Compliance). *P is* compliant *with Q (written $P \dashv Q$) iff $P \parallel Q \to^* P' \parallel Q' \not\to$ implies $P' = \mathbf{1}$.*

3 Contracts as games

We assume a denumerable universe of *events* $e, e', \ldots \in \mathbf{E}$, uniquely associated to *participants* $A, B, \ldots \in \mathbf{P}$ by a function $\pi : \mathbf{E} \to \mathbf{P}$. For all $A \in \mathbf{P}$, we write $\mathbf{E}_A$ for the set $\{e \in \mathbf{E} \mid \pi(e) = A\}$. For a sequence $\sigma = \langle e_0\, e_1 \cdots \rangle$ in E (possibly infinite), we write $\overline{\sigma}$ for the set of elements in σ; we write σ_i for the subsequence $\langle e_0 \cdots e_{i-1} \rangle$ containing exactly i events. If $\sigma = \langle e_0 \cdots e_n \rangle$, we write $\sigma\, e$ for the sequence $\langle e_0 \cdots e_n\, e \rangle$. The empty sequence is denoted by ε. For a set S, we denote with S^* the set of finite sequences over S, and with S^∞ the set of finite and infinite sequences over S.

A contract is modelled in [4] as a concurrent game featuring *obligations* (what I must do in a given state) and *objectives* (what I wish to obtain). Obligations are modelled as an event structure (ES).

Definition 3 (**Event structure [12]**). *An event structure $\mathcal{E}$ is a triple $\langle E, \#, \vdash \rangle$, where:*

- *E is a set of events,*

- $\# \subseteq E \times E$ *is an irreflexive and symmetric* conflict *relation. For a set of events* X*, the predicate* $CF(X)$ *is true iff* X *is* conflict-free*, i.e.* $CF(X) \triangleq (\forall e, e' \in X : \neg(e \# e'))$.
- $\vdash \subseteq \{X \subseteq_{fin} E \mid CF(X)\} \times E$ *is the* enabling *relation, which is* saturated*, i.e.:*

$$\forall X \subseteq Y \subseteq_{fin} E.\ X \vdash e \wedge CF(Y) \implies Y \vdash e$$

Intuitively, an enabling $X \vdash e$ models the fact that, if all the events in X have happened, then e is an obligation for $\pi(e)$. The conflict relation # is used to model non-deterministic choices: if $e \# e'$ then e and e' cannot occur in the same computation. An obligation may be discharged only by performing the required event, or any event in conflict with it. For instance, consider an internal choice between two events e_a and e_b. This can be modelled by an ES with enablings $\emptyset \vdash e_a$, $\emptyset \vdash e_b$ and conflict $e_a \# e_b$. After the choice (say, of e_a), the obligation e_b is discharged. The other component of a contract is a function Φ which associates each participant A with a set of sequences in $\mathbf{E}^\infty$ (the set of finite or infinite sequences on $\mathbf{E}$), which enumerates all the executions where A has a positive payoff.

Definition 4 (Contract). *A contract* $\mathcal{C}$ *is a pair* $\langle \mathcal{E}, \Phi \rangle$*, where:*

(a) $\mathcal{E} = \langle E, \#, \vdash, \ell \rangle$ *is a* labelled *event structure, with* $E \subseteq \mathbf{E}$ *and labelling function* $\ell : E \to \mathbf{A} \cup \overline{\mathbf{A}}$.

(b) $\Phi : \mathbf{P} \rightharpoonup \wp(E^\infty)$ *associates each participant with a set of traces.*

Note that Φ is a partial function (from $\mathbf{P}$ to sets of event traces), hence a contract is not supposed to define payoffs for all the participants in $\mathbf{P}$. Hereafter, we shall assume that if $\mathcal{C}$ prescribes for A some obligations, then $\mathcal{C}$ must also declare A's payoffs, i.e. we ask that $\Phi(\pi(e)) \neq \bot$ whenever $X \vdash e$ in $\mathcal{E}$.

Given two contracts $\mathcal{C}, \mathcal{C}'$, we denote with $\mathcal{C} \mid \mathcal{C}'$ their composition. If $\mathcal{C}$ is A's contract and $\mathcal{C}'$ is the contract of an adversary M of A, then a naïve composition could easily lead to an attack, e.g. M's contract could say that A must pay her 1M euros. To avoid such kinds of attacks, contract composition is a partial operation. We do *not* compose contracts which assign payoffs to the same participant.

Definition 5 (Contract composition). *We say that two contracts* $\mathcal{C} = \langle \mathcal{E}, \Phi \rangle$ *and* $\mathcal{C}' = \langle \mathcal{E}', \Phi' \rangle$ *are* composable *iff* $\forall A \in \mathbf{P}.\ (\Phi(A) = \bot \vee \Phi'(A) = \bot)$*. If* $\mathcal{C}$*,* $\mathcal{C}'$ *are composable, we define their composition as* $\mathcal{C} \mid \mathcal{C}' = \langle \mathcal{E} \sqcup \mathcal{E}', \Phi \sqcup \Phi' \rangle$*, where* $\mathcal{E} \sqcup \mathcal{E}' = \langle E \cup E', \# \cup \#', \vdash \cup \vdash', \ell \cup \ell' \rangle$.

A crucial notion on contracts is that of *agreement*. Intuitively, when A agrees on a contract $\mathcal{C}$, then she can safely initiate an interaction with the other participants, and be guaranteed that the interaction will not "go wrong" — even in the presence of attackers. This does not mean that A will always reach her objectives: we intend that A agrees on a contract when, in all the interactions where she does not succeed, then some other participant must be found dishonest. That is, we consider A satisfied if she can blame another participant. In real-world applications, a judge may provide compensations to A, or impose a punishment to the participant who has violated the contract.

We interpret a contract $\mathcal{C} = \langle \mathcal{E}, \Phi \rangle$ as a multi-player game, where the players concurrently perform events in order to reach the objectives in Φ. A *play* σ of $\mathcal{C}$ is a (finite or infinite) sequence of events of $\mathcal{E}$, such that each event e in σ is enabled by its predecessors. Formally, the plays of $\mathcal{E}$ are the traces of the labelled transition system $\mathsf{EvS}(\mathcal{E})$ induced by the relation $\mathcal{E} \xrightarrow{e} \mathcal{E}[e]$, where $\emptyset \vdash e$, and $\mathcal{E}[e]$ is the *remainder* of $\mathcal{E}$ after executing e.

Definition 6 (Remainder of an ES). *For all ES* $\mathcal{E} = \langle E, \#, \vdash, \ell \rangle$ *and for all* $e \in E$*, we define the ES* $\mathcal{E}[e]$ *as* $\langle E', \#', \vdash', \ell' \rangle$*, where:*

$$\begin{array}{rcl} E' & = & E \setminus (\{e\} \cup \{e' \mid e \# e'\}) \\ \# ' & = & \# \setminus (\{(e,e') \mid e \# e'\} \cup \{(e',e) \mid e \# e'\}) \\ \vdash' & = & \{(X \setminus \{e\}, e') \mid (X, e') \in \vdash''\} \ \textit{where} \ \vdash'' = \vdash \setminus \{(X, e') \mid e' \# e \vee e' = e \vee \neg CF(X \cup \{e\})\} \\ \ell' & = & \ell \setminus (\{(e', \ell(e')) \mid e' \# e\} \cup \{(e, \ell(e))\}) \end{array}$$

$$
\begin{array}{rcl}
[\![\mathbf{1}]\!]^A_\rho & = & \langle \{e\}, \emptyset, \{(\emptyset, e)\}, \{(e, \checkmark)\} \rangle \text{ where } e \in \mathbf{E}_A \\
[\![x]\!]^A_\rho & = & \rho(x) \;=\; \langle E, \#, \vdash, \ell \rangle \text{ where } E \subseteq \mathbf{E}_A \\
[\![\alpha.P]\!]^A_\rho & = & \langle E \cup \{e_\alpha\},\ \#,\ \vdash',\ \ell \cup \{(e_\alpha, \alpha)\} \rangle \text{ where } [\![P]\!]^A_\rho = \langle E,\ \#,\ \vdash,\ \ell \rangle,\ e_\alpha \in \mathbf{E}_A \setminus E, \text{ and} \\
 & & \vdash' = \{(\emptyset, e_\alpha)\} \cup \{(\{e_\alpha\} \cup X, e) \mid (X, e) \in \vdash\} \\
[\![\bigodot_{i \in I} P_i]\!]^A_\rho & = & \langle \bigcup E_i,\ \#,\ \bigcup \vdash_i,\ \bigcup \ell_i \rangle \text{ where } [\![P_i]\!]^A_\rho = \langle E_i,\ \#_i,\ \vdash_i,\ \ell_i \rangle,\ E_i \text{ pairwise disjoint, and} \\
 & & \# \;=\; \bigcup \#_i \cup \{(e, e') \mid (\emptyset, e) \in \vdash_i \wedge (\emptyset, e') \in \vdash_j \wedge i \neq j\}, \text{ with } \bigodot \in \{\sum, \bigoplus\} \\
[\![rec\, x.\, P]\!]^A_\rho & = & fix\, \Gamma \text{ where } \Gamma(\mathcal{E}) = [\![P]\!]^A_{\rho\{\mathcal{E}/x\}} \\
[\![P_1 \,\|\, P_2]\!]^{A_1 A_2}_\rho & = & \langle E,\ \#_1 \cup \#_2,\ \vdash,\ \ell \rangle \text{ where } [\![P_i]\!]^{A_i}_\rho = \langle E_i,\ \#_i,\ \vdash_i,\ \ell_i \rangle,\ E = E_1 \cup E_2 \text{ with } E_1 \cap E_2 = \emptyset,\ \ell = \ell_1 \cup \ell_2, \\
 & & \vdash \;=\; \{(X \cup Y, e) \mid \ell(e) \in \overline{\mathbf{A}} \wedge (X, e) \in \vdash_i \wedge \forall e' \in X. \exists e'' \in Y. e'' \in E \setminus E_i \wedge \ell(e') = \overline{\ell(e'')}\} \\
 & & \quad \cup\ \{(X \cup Y \cup \{\hat{e}\}, e) \mid \ell(e) \in \mathbf{A} \wedge (X, e) \in \vdash_i \wedge \forall e' \in X.\ \exists e'' \in Y.\ \big(\\
 & & \qquad\qquad e'' \in E \setminus E_i \wedge \ell(e') = \overline{\ell(e'')} \wedge \hat{e} \in E \setminus E_i \wedge \ell(\hat{e}) = \overline{\ell(e)} \big)\}
\end{array}
$$

Figure 2: Denotational semantics of session types.

A *strategy* Σ for A is a function which associates to each finite play σ a set of events of A (possibly empty), such that if $e \in \Sigma(\sigma)$ then σe is still a play. A play $\sigma = \langle e_0 e_1 \cdots \rangle$ *conforms* to a strategy Σ for A when, for all $i \geq 0$, $e_i \in \mathbf{E}_A$ implies $e_i \in \Sigma(\sigma_i)$. A play is *fair* w.r.t. a strategy Σ iff any event permanently prescribed by Σ is eventually performed.

Definition 7 (Fair play). *A play $\sigma = \langle e_0 e_1 \cdots \rangle$ is* fair *w.r.t. the strategy Σ iff:*

$$\forall i \leq |\sigma|.\ \big(\forall j : i \leq j \leq |\sigma|.\ e \in \Sigma(\sigma_j)\big) \implies \exists h \geq i.\ e_h = e$$

A participant A is *innocent* in a play if A has no persistently enabled events, i.e. if all her enabled events are either performed or conflicted.

Definition 8 (Innocence). *We say A* innocent in σ *iff* $\forall i \geq 0. \forall e \in \mathbf{E}_A.\ (\overline{\sigma_i} \vdash e \implies \exists j \geq i.\ e_j \# e \vee e_j = e)$. *If A is not innocent in σ, then we say she is* culpable.

We now define when a participant *wins* in a play. If A is culpable, then she loses. If A is innocent, but some other participant is culpable, then A wins. Otherwise, if all participants are innocent, then A wins if she has a positive payoff in the play. Formally, σ is a winning play of A iff $\sigma \in \mathcal{W}A$, defined below.

Definition 9 (Winning plays and strategies). *We define the function $\mathcal{W} : \mathbf{P} \to \wp(\mathbf{E}^\infty)$ as follows:*

$$\mathcal{W}A \;= \{\sigma \in \Phi A \mid \forall B : B \text{ innocent in } \sigma\} \cup \{\sigma \mid A \text{ innocent in } \sigma, \text{ and } \exists B \neq A : B \text{ culpable in } \sigma\}$$

We say that Σ is winning *for A in $\mathcal{C}$ iff A wins in every fair play of $\mathcal{C}$ which conforms to Σ.*

Intuitively, A agrees with $\mathcal{C}$ when she has a strategy Σ which allows her to win in all fair plays conform to Σ. Note that neglecting unfair plays is quite reasonable: indeed, an unfair scheduler could easily prevent an honest participant (ready to fulfil all her obligations) from performing any action.

Definition 10 (Agreement). *A participant A* agrees *on $\mathcal{C}$ whenever A has a winning strategy in $\mathcal{C}$.*

4 Compliance as agreement

We now relate session types with contracts. To do that, we start by introducing an event structure semantics for session types. This denotational semantics is then related to a turn-based operational semantics

of session types (Figure 3), which preserves the notion of compliance (Lemma 14). In Definition 15 we transform session types into contracts. Theorem 16 establishes a correspondence between compliance of session types and winning strategies in contracts.

Definition 11 (ES semantics of session types). *The denotation of session types is defined by the rules in Figure 2, where ρ is an environment mapping variables x to ESs.*

The denotation of session types is almost straightforward. Note that the parameter A is used to associate all the events of the constructed ES to that participant. The enabling relation of compositions of session types takes into account the different flavour of the events (actions) involved. Intuitively, an action $\overline{b}$ in a contract such as $\overline{a}.\overline{b}$ must wait for its prefix $\overline{a}$, and for a matching a-synchronized action. On the other hand, an action in $\mathbf{A}$ such as c must also wait to be matched by a synchronizing action $\overline{c}$. This behaviour is simulated in the event structure of the contract: for an enabling $X \vdash e$ with $\ell(e) \in \overline{\mathbf{A}}$, we add to X the set Y of all the matching events of X. Instead, for an enabling $X \vdash e$ with $\ell(e) \in \mathbf{A}$, we also add to X and Y the coaction of $\ell(e)$.

As session types have recursion, the standard machinery on fixed points is needed. Henceforth, following closely what is done in [12], we introduce a notion of partial ordering on event structures. The intuition is that $\mathcal{E}$ is less or equal to $\mathcal{E}'$ whenever each configuration of the former is a configuration of the latter, and each configuration of $\mathcal{E}'$ where the events are those of $\mathcal{E}$ is a configuration of $\mathcal{E}$ as well.

Definition 12 (Ordering of ESs). *Let $\mathcal{E} = \langle E, \#, \vdash, \ell \rangle$ and $\mathcal{E}' = \langle E', \#', \vdash', \ell' \rangle$ be two ESs. Then we write $\mathcal{E} \trianglelefteq \mathcal{E}'$ iff:*

- *$E \subseteq E'$, $\# \subseteq \#'$, $\vdash \subseteq \vdash'$ and $\forall e \in E.\ \ell'(e) = \ell(e)$,*
- *for all $e_1, e_2 \in E$, if $(e_1, e_2) \in \#'$ then $(e_1, e_2) \in \#$, and*
- *for all $X \subseteq E$, $e \in E$, if $(X, e) \in \vdash'$ then $(X, e) \in \vdash$.*

The relation $\trianglelefteq$ is a partial order on event structures. An ω-chain of ESs $\mathcal{E}_1 \trianglelefteq \mathcal{E}_2 \trianglelefteq \cdots \trianglelefteq \mathcal{E}_n \trianglelefteq \ldots$ has a least upper bound defined as $\bigsqcup \mathcal{E}_i = (\bigcup_i E_i, \bigcup_i \#_i, \bigcup_i \vdash_i, \bigcup_i \ell_i)$. The ES $\underline{\emptyset} = \langle \emptyset, \emptyset, \emptyset, \emptyset \rangle$ is the least element of the partial order. Given a unary operator $\mathbf{F}$ on event structures, we say that it is *continuous on events* iff for every ω-chain of ESs $\mathcal{E}_1 \trianglelefteq \mathcal{E}_2 \trianglelefteq \cdots \trianglelefteq \mathcal{E}_n \trianglelefteq \ldots$ it holds that $\mathbf{F}(\bigcup_i E_i) = \bigcup_i \mathbf{F}(E_i)$. If furthermore the operator $\mathbf{F}$ is monotonic with respect to $\trianglelefteq$ then $\mathbf{F}$ is *continuous*. Given a continuous unary operator $\mathbf{F}$, we can then define its fixed point standardly using Tarski's theorem, as event structures with $\trianglelefteq$ are a complete partial order with bottom. The fixed point is denoted by $\mathit{fix}\ \Gamma = \bigsqcup \mathbf{F}(\underline{\emptyset})$. It is standard to prove that the operators defined by the denotational semantics in Figure 2 are continuous.

We shall now relate the denotational semantics in Definition 11 with an operational semantics of binary session types where the two participants alternate in firing actions (Figure 3). To do that, we extend the syntax of session types with the term $[\overline{a}]P$, where $[\overline{a}]$ models a one-position buffer storing $\overline{a}$. Also, we tacitly assume unfolding of recursion. A participant with an internal choice $\overline{a}.P \oplus Q$ can fire the action $\overline{a}$ (if the buffer is empty), and write $\overline{a}$ to the buffer. The next turn is of the other participant, which can empty the buffer by firing a in an external choice. To be coherent with the event structure semantics, we also assume that the success state $\mathbf{1}$ fires an action $\checkmark \in \overline{\mathbf{A}}$ before reaching the stuck state $\mathbf{0}$.

The following theorem relates the denotational and the turn-based operational semantics of session types. Their (action-labelled) LTSs are strongly bisimilar. Below, we denote with $\mathsf{ES}(\mathcal{E})$ the transition systems induced by the relation $\mathcal{E} \xrightarrow{e} \mathcal{E}'$, by relabelling transitions with actions $\ell(e)$, and we denote with $\mathsf{TS}(P)$ the labelled transition system induced by the turn-based relation $\twoheadrightarrow$.

Theorem 13. *For all session types P, Q, we have $\mathsf{TS}(P \,\|\, Q) \sim \mathsf{ES}(\llbracket P \,\|\, Q \rrbracket)$.*

The turn-based semantics of session types preserves the compliance relation of Definition 2.

$$(\overline{a}.P \oplus Q) \parallel R \xrightarrow{\overline{a}}\!\!\!\!\rightarrow [\overline{a}]P \parallel R \qquad (a.P + Q) \parallel [\overline{a}]R \xrightarrow{a}\!\!\!\!\rightarrow P \parallel R \qquad \mathbf{1} \parallel P \xrightarrow{\checkmark}\!\!\!\!\rightarrow \mathbf{0} \parallel P$$

Figure 3: Turn-based operational semantics of session types (symmetric rules omitted).

Lemma 14. *$P \dashv Q$ iff $P \parallel Q \twoheadrightarrow^* P' \parallel Q' \not\twoheadrightarrow$ implies $P' = \mathbf{0}$.*

We now define a transformation from session types P to contracts, denoted by $\mathcal{C}_A(P)$. The parameter A is used to properly assign the obligations and the objective to participant A.

Definition 15 (Contract of a session type). *For all session types P and participants A, we define the contract $\mathcal{C}_A(P)$ as $\langle [\![P]\!]^A_\emptyset, \Phi \rangle$, where $\Phi A = \{\sigma \in \mathbf{E}^\infty \mid \sigma \in \mathbf{E}^* \implies \exists e \in \overline{\sigma} \cap \mathbf{E}_A : \ell(e) = \checkmark\}$.*

We now establish a correspondence between compliance in session types and the existence of certain winning strategies in contracts. To do that, we consider strategies which ensure A to be innocent in every (fair) play. The greatest of such strategies is the *eager strategy* $\Sigma_A(\sigma) = \{e \in \mathbf{E}_A \mid \overline{\sigma} \vdash e\}$ which prescribes A to do all her enabled events. The session type P (say, of participant A) is compliant with Q iff the eager strategy is winning for A in the contract $\mathcal{C}_A(P) \mid \mathcal{C}_B(Q)$.

Theorem 16. *$P \dashv Q$ iff the eager strategy is winning for A in $\mathcal{C}_A(P) \mid \mathcal{C}_B(Q)$.*

By the theorem above, it follows that compliance implies agreement.

Corollary 17. *If $P \dashv Q$, then A agrees on $\mathcal{C}_A(P) \mid \mathcal{C}_B(Q)$.*

Note that the converse implication does *not* hold: for instance, for $P = \overline{a}.\overline{c} \oplus \overline{b}$ and $Q = a + b$, we have that $P \not\dashv Q$, but A agrees on $\mathcal{C}_A(P) \mid \mathcal{C}_B(Q)$. Indeed, choosing the branch $\overline{b}$ leads to a winning strategy for A. Note instead that P is *not* weakly compliant with Q according to [11], because no orchestrator can prevent A from choosing the branch $\overline{a}$. However, $P' = \overline{a}.\overline{c} + \overline{b}$ is weakly compliant with Q, because the orchestrator can resolve the external non-determinism by choosing the branch $\overline{b}$. Weak compliance can be formalised in game-based contracts by modelling the orchestrator as a third player of the game (who can use *any* strategy to favour the interaction between A and B), and by adapting the construction of the contracts to take into account for the moves of the orchestrator.

An example. We now illustrate with the help of an example the transformation from session types to game-based contracts. Below, we use the following shorthands: $a \vdash b$ for $\{a\} \vdash b$, and $\vdash e$ for $\emptyset \vdash e$.

Consider two participants A and B, with session types $P = \overline{a} \oplus \overline{b}.\overline{a}$ and $Q = a.b + b.a + c$, respectively (trailing $\mathbf{1}$s are omitted). According to Definition 2, the session type of A is compliant with that of B, while the converse does not hold. Below we construct the event structures associated to P and Q, and the one associated to the their composition $P \parallel Q$. To ease the reading, we decorate actions in P and Q with the events they will be associated with in the event structures; we stipulate that the events of A have odd indexes, whereas those of B have even ones. Hence, we have:

$$P = \overline{a}_{e_1}.\mathbf{1}_{e_3} \oplus \overline{b}_{e_5}.\overline{a}_{e_7}.\mathbf{1}_{e_9} \quad \text{and} \quad Q = a_{e_2}.b_{e_4}.\mathbf{1}_{e_6} + b_{e_8}.a_{e_{10}}.\mathbf{1}_{e_{12}} + c_{e_{14}}.\mathbf{1}_{e_{16}}$$

By the construction in Def. 11, we have:

$$[\![P]\!]^A_\rho = (\{e_1, e_3, e_5, e_7, e_9\}, \quad \{e_1 \# e_5\}, \quad \{\vdash e_1, \vdash e_5, e_1 \vdash e_3, e_5 \vdash e_7, e_7 \vdash e_9\}, \quad \ell_P)$$

where $\ell_P(e_1) = \overline{a}$, $\ell_P(e_5) = \overline{b}$, $\ell_P(e_7) = \overline{a}$, and the others are labelled with $\checkmark$. Furthermore:

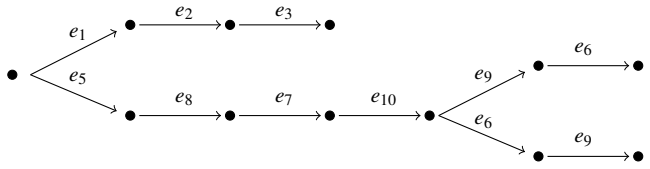
$$[\![Q]\!]^B_\rho = (\{e_2, e_4, e_6, e_8, e_{10}, e_{12}, e_{14}, e_{16}\},\ \{e_2 \# e_8, e_2 \# e_{14}, e_8 \# e_{14}\},\ \left\{ \begin{array}{l} \vdash e_2, \vdash e_8, \vdash e_{14}, e_2 \vdash e_4, e_4 \vdash e_6 \\ e_8 \vdash e_{10}, e_{10} \vdash e_{12}, e_{14} \vdash e_{16} \end{array} \right\},\ \ell_Q)$$

where $\ell_Q(e_2) = a = \ell_Q(e_{10})$, $\ell_Q(e_4) = b = \ell_Q(e_8)$, $\ell_Q(e_{14}) = c$, and the other events are labelled with $\checkmark$. The event structure associated to $P \parallel Q$ is:

$$[\![P \parallel Q]\!]_{\emptyset}^{A,B} = (E_P \cup E_Q, \quad \#_P \cup \#_Q, \quad \vdash, \quad \ell_P \cup \ell_Q) \qquad \text{where:}$$

$$\begin{array}{rcl} \vdash & = & \vdash e_1, \vdash e_5, \{e_1,e_2\} \vdash e_3, \{e_1,e_{10}\} \vdash e_3, \{e_5,e_8\} \vdash e_7, \{e_5,e_4\} \vdash e_7, \{e_2,e_7\} \vdash e_9, \{e_7,e_{10}\} \vdash e_9, \\ & & e_1 \vdash e_2, e_7 \vdash e_2, \{e_1,e_2,e_5\} \vdash e_4, \{e_7,e_2,e_5\} \vdash e_4, \{e_4,e_5\} \vdash e_6, e_5 \vdash e_8, \{e_8,e_5,e_1\} \vdash e_{10}, \\ & & \{e_8,e_5,e_7\} \vdash e_{10}, \{e_{10},e_1\} \vdash e_{12}, \{e_{10},e_7\} \vdash e_{12} \end{array}$$

The event-labelled transition system of $[\![P \parallel Q]\!]$ and the eager strategy Σ_A of A are depicted below:



$$\Sigma_A(\sigma) = \begin{cases} \{e_1,e_5\} & \text{if } \overline{\sigma} = \emptyset \\ \{e_3\} & \text{if } e_2 \in \overline{\sigma} \\ \{e_7\} & \text{if } e_8 \in \overline{\sigma} \\ \{e_9\} & \text{if } e_{10} \in \overline{\sigma} \\ \emptyset & \text{otherwise} \end{cases}$$

We can see that A wins in all the fair plays which conform to the eager strategy Σ_A. Since Σ_A is winning, then A agrees on $\mathcal{C}_A(P) \mid \mathcal{C}_B(Q)$. Then, by Theorem 16, $P \dashv Q$.

On the contrary, we notice that B has no winning strategies: indeed, whenever A chooses to perform event e_1, then B is obliged to fire e_2 to recover his innocence, and then he gets stuck (and non-successful) when A fires e_3. Then, by Theorem 16 it follows that $Q \not\dashv P$.

5 Conclusions

We have related the notion of compliance in binary session types with the one of agreement in game-based contracts. In particular, we have shown that two session types are compliant if and only if their encodings in game-based contract admit an agreement via a winning *eager* strategy (Theorem 16).

A relevant question is whether non-eager strategies are meaningful to define weaker notions of compliance for session types. This mostly depends on the interpretation of the internal choice operator $\oplus$. The usual meaning of an internal choice $\overline{a} \oplus \overline{b}$ of a participant A is that A is willing to opt between the two choices, and *both* of them must be available as external choices of the other participant B.

Just to give a more realistic flavour to our scenario, assume that B is a bartender which only accepts payments in cash, while A is a customer willing to pay either by cash or by credit card. Under the standard notion of compliance, the two session types:

$$P_A \;=\; \overline{payCash} \oplus \overline{payCC} \qquad\qquad P_B \;=\; payCash$$

are *not* compliant, and so (by Theorem 16) the eager strategy is *not* winning in $\mathcal{C}_A(P_A) \mid \mathcal{C}_B(P_B)$.

A different interpretation of the internal choice of A would be the following: A is willing to choose between $\overline{payCash}$ and $\overline{payCC}$ if both options are available, but she will also accept to pay cash (resp. to pay by credit card) if this is the only option available. This interpretation is coherent with the fact that the contract $\mathcal{C}_A(P_A) \mid \mathcal{C}_B(P_B)$ admits an agreement, via a non-eager strategy which requires A to renounce to the $\overline{payCC}$ alternative.

Similarly, we expect that other interpretations of compliance for session types (e.g. that in [8, 6], where internal *vs.* internal choices and external *vs.* external choices may be compliant, in some cases) can be related to game-based agreements, via suitable (sub)classes of strategies.

Acknowledgments. This work has been partially supported by Aut. Reg. of Sardinia grants L.R.7/2007 CRP-17285 (TRICS) and P.I.A. 2010 ("Social Glue"), by MIUR PRIN 2010-11 project "Security Horizons", and by EU COST Action IC1201 "Behavioural Types for Reliable Large-Scale Software Systems" (BETTY).

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] Franco Barbanera & Ugo de'Liguoro (2010): *Two notions of sub-behaviour for session-based client/server systems*. In: *PPDP*, pp. 155–164, doi:10.1145/1836089.1836109.

[3] Franco Barbanera & Ugo de'Liguoro (2014): *Loosening the notions of compliance and sub-behaviour in client/server systems*. In: *Proc. ICE*. Available at `http://arxiv.org/abs/1311.5802`.

[4] Massimo Bartoletti, Tiziana Cimoli & Roberto Zunino (2013): *A theory of agreements and protection*. In: *Proc. POST*, *LNCS* 7796, Springer, pp. 186–205, doi:10.1007/978-3-642-36830-1_10.

[5] Massimo Bartoletti, Alceste Scalas & Roberto Zunino (2014): *A semantic deconstruction of session types*. In: *Proc. CONCUR*, pp. 402–418.

[6] Massimo Bartoletti, Emilio Tuosto & Roberto Zunino (2012): *On the Realizability of Contracts in Dishonest Systems*. In: *Proc. COORDINATION*, pp. 245–260, doi:10.1007/978-3-642-30829-1_17.

[7] 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.

[8] 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.

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

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

[11] Luca Padovani (2010): *Contract-based discovery of Web services modulo simple orchestrators*. *Theor. Comput. Sci.* 411(37), pp. 3328–3347, doi:10.1016/j.tcs.2010.05.002.

[12] Glynn Winskel (1986): *Event Structures*. In: *Advances in Petri Nets*, doi:10.1007/3-540-17906-2_31.