Contracts as games on event structures

Massimo Bartoletti*, Tiziana Cimoli, G. Michele Pinna

Dipartimento di Matematica e Informatica, Università degli Studi di Cagliari, Italy

Roberto Zunino

Dipartimento di Matematica, Università degli Studi di Trento, Italy

Abstract

Event structures are one of the classical models of concurrent systems. The idea is that an *enabling* $X \vdash e$ represents the fact that the event e can only occur after all the events in the set X have already occurred. By interpreting events as actions promised by some participants, and by associating each participant with a goal (a function on sequences of events), we use event structures as a formal model for *contracts*. The states of a contract are sequences of events; a participant has a contractual obligation (in a given state) whenever some of its events is enabled in such a state. To represent the fact that participants are mutually distrusting, we study concurrent games on event structures; there, participants may play by firing events in order to reach their goals, and eventually win, lose or tie.

A crucial notion arising in this setting is that of *agreement*: a participant agrees on a set of contracts if she has a strategy to reach her goals in all the plays conforming to her strategy (or to make another participant sanctionable for not honouring an obligation). Another relevant notion is *protection*: a participant is protected by her contract when she has a strategy to avoid losing in any contexts, even in those where she has not reached an agreement. We study conditions for obtaining agreement and protection, and we show that these properties mutually exclude each other in a certain class of contracts. We then relate the notion of agreement in contracts with that of *compliance* in session types. In particular, we show that compliance corresponds to the fact that *eager* strategies lead to agreement.

Keywords: contracts, event structures, compliance, session types

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).

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 [1, 2]), or which ensures the possibility of

☆Work partially supported by Aut. Region of Sardinia under grants L.R.7/2007 CRP-17285 (TRICS), P.I.A. 2010 Project "Social Glue", by MIUR PRIN 2010-11 project "Security Horizons", and by EU COST Action IC1201 (BETTY).

* *Corresponding author.* Dipartimento di Matematica e Informatica, Università degli Studi di Cagliari, via Ospedale 72, 09124 Cagliari (Italy), e-mail: bart@unica.it

always reaching success [3, 4]. Weaker variants of compliance allow services to discard some messages [5], or involve orchestrators which can sometimes rearrange them [6]. All these approaches express contracts as terms of some process calculus.

In this paper we study compliance in the semantic setting of event structures (ES [7]). 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 calculi for concurrency.

In our setting, a contract specifies the behaviour promised and expected by a participant or set of participants. Contracts coming from different participants can be composed together. In our view, *agreement* (a generalisation of compliance) is a property of composed contracts, which — roughly — ensures an acceptable interaction to each participant in the composition.

Our contracts are built upon four principal notions:

Events are the atomic observables. For instance, "Alice gives an apple to Bob" can be modelled as an event (say, a) in an ES. We assume that each event is unique, i.e. it cannot occur twice in the same computation. Thus, if Alice has to give two apples to Bob, we assume two events a_0, a_1 (representing two distinct occurrences of the same action).

Participants are the entities which advertise contracts, and are bound to perform the events prescribed by their contracts. We assume that each event is associated with a unique participant. For instance, if both Alice and Carol have to give an apple to Bob, we use two distinct events.

Obligations make explicit the causal dependencies between the events performed by participants. For instance, Alice's contract clause "I will give an apple to Bob after I have received a banana" induces an obligation for her to do event a after event b has been performed, since she has promised to do it. Event structures are a natural model for obligations; for instance, we can interpret the above clause as the *enabling* $\{b\} \vdash a$.

Objectives express the degree of "satisfaction" of a participant in a contract execution. Contracts associate each participant to an objective function, which in turn associates each execution with a *payoff*, which can be "win", "lose", or "tie".

In the above setting, we provide a formal definition of contracts, by interpreting their semantics as a multi-player concurrent game on event structures. We then formalise two key notions about contracts, namely *agreement* and *protection*. Intuitively, agreement is a property of a contract which results from the composition of a number of individual contracts from a set of participants. A participant agrees with such composed contract if she has a strategy to interact with the other participants so that in each interaction she either wins, or it is possible to blame another participant who is not honouring his obligations. Instead, protection is a property of a contract of a single participant. It requires that, whenever the contract is composed with any other contracts, possibly crafted by adversaries, then the participant has a strategy to avoid losing (by instead winning or tying) in the interactions with such adversaries.

Contributions. The main contributions of this paper are the following:

- We provide a formal definition for the intuitive notion of *agreement*. We study conditions for reaching agreements in contracts with *Offer-Request* payoffs, where participants request some actions in exchange for an offered service. Lemma 4.17 gives a necessary condition for agreement, while Theorem 4.19 gives a sufficient one.
- We interpret binary session types [8, 9] as contracts, by providing them with event structure semantics (Definitions 5.12 and 5.19). We establish our semantics faithful to the original one, by proving that the associated event structure is bisimilar to the session type in its operational semantics (Theorems 5.17 and 5.20). We then exploit this correspondence result to show that compliance in session types holds whenever eager strategies lead to an agreement in the associated contract (Theorem 5.23). To

prove this correspondence, we establish an auxiliary result about event structures. We provide them with two notions of Labelled Transition Systems, one based on the remainder, and the other one on configurations, and we relate them by bisimilarity (Lemma A.5).

- We formalise the notion of protection, and we study necessary conditions (Lemma 6.6) and sufficient conditions (Theorem 6.7) for obtaining protection in contracts with Offer-Request payoffs. We then show that agreement and protection are mutually exclusive for contracts with Offer-Request payoffs suffering from a circularity condition (Theorem 6.11). Roughly, the problem is that when the offers of the participants mutually depend on their requests, either there is a participant willing to perform the first offer, and so giving up protection, or each participant wants someone else to move first, so preventing an agreement to be reached.

The proofs of all our statements are provided either in the main text, or in Section A.

2. Event structures

Event structures (ES) are a model for concurrency introduced in [10]; they describe a process as performing events as time goes on. An *event* is a particular occurrence of an *action*, and different events may be occurrences of the same action. Each event e is labelled with the action $\ell(e)$ it is associated with. For instance, pressing n times a certain button is represented by a sequence of n distinct events, all with the same label. ESs are equipped with an *enabling* relation (written $\vdash$) to model *causality*, and a *conflict* relation (written $\#$) to model *non-determinism*. The enabling $X \vdash e$ models the fact that event e can be fired after all the events in X have been fired. A conflict $a\#b$ models that a and b cannot both occur in the same computation.

In this section we report some basic definitions and results about ES, which will be needed in our later technical development. Furthermore, we study labelled transition systems (LTS) over ES, which will be used to define plays in contracts.

2.1. Basic definitions

We assume a denumerable universe of *events* $\mathbf{E}$, ranged over by $a, b, e \ldots$, and a universe of *action labels* $\mathtt{A}$, ranged over by $\alpha, \beta, \ldots$.

Definition 2.1 (Conflict-free and consistent sets [11]). *Given a set of events $E \subseteq \mathbf{E}$ and a relation $\# \subseteq E \times E$, we define the predicate CF on sets $X \subseteq E$ as follows:*

$$CF(X) \;=\; \forall e, e' \in X : \neg(e \# e')$$

When $CF(X)$, we say that X is conflict-free*. We define the set Con of finite conflict-free sets as follows:*

$$Con \;=\; \{X \subseteq_{fin} E \mid CF(X)\}$$

Definition 2.2 (Event structure [7]). *An event structure is a 4-tuple $\mathcal{E} = (E, \#, \vdash, \ell)$, where*

- $E \subseteq \mathbf{E}$,
- $\# \;\subseteq E \times E$ *is an irreflexive and symmetric relation, called* conflict relation*,*
- $\vdash \subseteq Con \times E$ *is a relation, called* enabling relation*. We assume* $\vdash$ saturated*, i.e. $sat(\vdash) = \vdash$, where:*

$$sat(\vdash) = \{(Y, e) \mid (X, e) \in \vdash \text{ and } X \subseteq Y \in Con\}$$

- $\ell : E \to \mathtt{A}$ *is a* labelling function*.*

We say that $\mathcal{E}$ is finite *when E is finite; we say that $\mathcal{E}$ is* conflict-free *when the conflict relation is empty. We denote with* **ES** *the class of all event structures.*

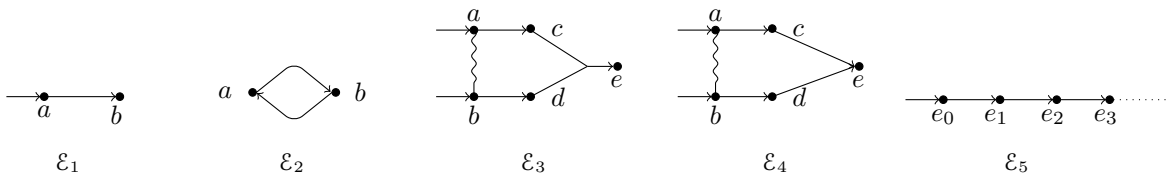


Figure 1: Graphical representation of ESs. An edge from node a to node b denotes an enabling $\{a\} \vdash b$. A conflict $a\#b$ is represented by a wavy line between a and b. We will use hyperdges to represent enablings of the form $X \vdash e$, when X is not a singleton (for instance, $\{c, d\} \vdash e$ in $\mathcal{E}_3$).

Definition 2.3 (Union of ESs). *Let $\mathcal{E} = (E, \#, \vdash, \ell)$ and $\mathcal{E}' = (E', \#', \vdash', \ell')$ be two* ES*s, with $\ell(e) = \ell'(e)$ for all $e \in E \cap E'$. We define their union $\mathcal{E} \sqcup \mathcal{E}'$ as $= (E \cup E', \vdash \cup \vdash', \# \cup \#', \ell \cup \ell')$.*

Notation 2.4 (Sequences of events). *We denote with $\langle e_0\, e_1 \ldots \rangle$ the (possibly infinite) sequence of events $e_0, e_1, \ldots$, and with E^∞ the set of such sequences over E. We use metavariables $\sigma, \eta, \ldots$ to range over such sequences. For a sequence $\sigma = \langle e_0\, e_1 \ldots \rangle$, we write: $\overline{\sigma}$ for the set of events in σ, and σ_i for the subsequence $\langle e_0 \ldots e_{i-1} \rangle$ containing the first i events of σ. If $\sigma = \langle e_0 \ldots e_n \rangle$ is finite, we write σe for the sequence $\langle e_0 \ldots e_n\, e \rangle$. The empty sequence is denoted by ε.*

Notation 2.5 (Shorthands). *We adopt the following conventions: (i) we write $e \in \mathcal{E}$ to mean that e is an event of $\mathcal{E}$; (ii) $\vdash e$ is a shorthand for $\emptyset \vdash e$; (iii) $a \vdash b$ is a shorthand for $\{a\} \vdash b$; (iv) for finite, conflict free sets $X, Y \subseteq E$, the enabling $X \vdash Y$ means that $\forall e \in Y.\ X \vdash e$.*

A *configuration* $C \subseteq E$ is a "snapshot" of the computation of a process: a set of events C (possibly infinite) is a configuration whenever for each event $e \in C$ we can find a finite sequence of events containing e, which is closed under the enabling relation.

Definition 2.6 (Configuration [7]). *For an* ES *$\mathcal{E} = (E, \#, \vdash, \ell)$, we say that $C \subseteq E$ is a* configuration of $\mathcal{E}$ *whenever $CF(C)$, and*

$$\forall e \in C.\ \exists \sigma = \langle e_0, \ldots, e_n \rangle.\ e \in \overline{\sigma} \subseteq C\ \wedge\ \forall i \leq n.\ \overline{\sigma_i} \vdash e_i$$

The set of all configurations of $\mathcal{E}$ is denoted by $\mathcal{F}_\mathcal{E}$.

Example 2.7. *Consider the five* ES*s in Figure 1. We have that:*

- *$\mathcal{E}_1$ has enablings $\vdash a$ and $a \vdash b$, and we have $\mathcal{F}_{\mathcal{E}_1} = \{\emptyset, \{a\}, \{a, b\}\}$.*
- *$\mathcal{E}_2$ has enablings $a \vdash b$ and $b \vdash a$, and we have $\mathcal{F}_{\mathcal{E}_2} = \{\emptyset\}$.*
- *$\mathcal{E}_3$ has enablings $\vdash a$, $\vdash b$, $a \vdash c$, $b \vdash d$, and $\{c, d\} \vdash e$, and the conflict $a\#b$. The configurations of $\mathcal{E}_3$ are $\emptyset$, $\{a\}$, $\{b\}$, $\{a, c\}$ and $\{b, d\}$. Note that no configuration contains e, because of the conflict $a\#b$.*
- *$\mathcal{E}_4$ has enablings $\vdash a$, $\vdash b$, $a \vdash c$, $b \vdash d$, $c \vdash e$ and $d \vdash e$. The configurations of $\mathcal{E}_4$ are $\emptyset$, $\{a\}$, $\{b\}$, $\{a, c\}$, $\{b, d\}$, $\{a, c, e\}$ and $\{b, d, e\}$.*
- *$\mathcal{E}_5$ has configurations $\bigcup_{i<k}\{e_i\}$, for all k. Also, the infinite set $\bigcup_{i\in\mathbb{N}} e_i$ is a configuration of $\mathcal{E}_5$.* □

Lemma 2.8. *If C is a finite nonempty configuration of $\mathcal{E}$, there exists some $e \in C$ such that $C \setminus \{e\} \in \mathcal{F}_\mathcal{E}$.*

Proof. By induction on $|C|$. For $|C| \in \{0, 1\}$, trivial. Otherwise, by *coincidence freeness* (Theorem 1.1.9 in [7]), we have some $Y \in \mathcal{F}_\mathcal{E}$ with $Y \subset C$. By Lemma 1.1.11 in [7], and by the finiteness of C, we can build a sequence of configurations:

$$Y\ \subseteq\ Y \cup \{e_1\}\ \subseteq\ Y \cup \{e_1, e_2\}\ \subseteq\ \cdots\ \subseteq\ Y \cup \{e_1, \ldots, e_m\}\ \subseteq\ C$$

by adding a new event at each step. Hence, $C = (Y \cup \{e_1, \ldots, e_m\}) \setminus \{e_m\}$, and so there exists $e = e_m$ such that $C \setminus \{e\} \in \mathcal{F}_\mathcal{E}$. □

2.2. Labelled Transition Systems over configurations

We now define, for each ES, an LTS over its configurations. The states of this LTS, which we denote with $\rightarrow_{\mathcal{E}}$ in Definition 2.9, are the finite configurations of $\mathcal{E}$; a transition with label e from state C exists whenever e is enabled by (and not in conflict with) C in $\mathcal{E}$.

Definition 2.9 (LTS over configurations). *For all event structures $\mathcal{E} = (E, \#, \vdash, \ell)$, we define the LTS $(\wp_{fin}(E), E, \rightarrow_{\mathcal{E}})$ as follows:*

$$C \xrightarrow{e}_{\mathcal{E}} C \cup \{e\} \qquad \text{if } C \vdash e,\ e \notin C \text{ and } CF(C \cup \{e\})$$

Notation 2.10. *As usual, we overload the symbol $\rightarrow$ to denote both the LTS and its transition relation. To make explicit that some state q belongs to the LTS $\rightarrow$, we write the state as a pair $(q, \rightarrow)$.*

Further, for each LTS $(\mathcal{Z}, E, \rightarrow)$ and each function $\ell : E \to \mathtt{A}$, we define the LTS $(\mathcal{Z}, \mathtt{A}, \rightarrow^{\ell})$ with the following transition relation:

$$q \xrightarrow{\ell(e)}^{\ell} q' \qquad \text{whenever } q \xrightarrow{e} q'$$

The following lemma states that a transition labelled e and enabled in a (finite) configuration C, will also be enabled in all the supersets of C not in conflict with e.

Lemma 2.11. *For all $C, C' \subseteq_{fin} E$, and for all $e \in E$ such that $CF(C' \cup \{e\})$:*

$$C \xrightarrow{e}_{\mathcal{E}} \ \wedge\ C \subseteq C' \ \wedge\ e \notin C' \implies C' \xrightarrow{e}_{\mathcal{E}}$$

Proof. By Definition 2.9, $C \xrightarrow{e}_{\mathcal{E}}$ implies that $C \vdash e$. By saturation (since $CF(C')$), this in turn implies that $C' \vdash e$. Therefore, since $e \notin C'$ and $CF(C' \cup \{e\})$ then we conclude that $C' \xrightarrow{e}_{\mathcal{E}}$. □

We establish a further auxiliary result, which relates (possibly infinite) configurations of $\mathcal{E}$ with the states of the LTS $\rightarrow_{\mathcal{E}}$. A direct consequence is that the states in the LTS reachable from the initial state $\emptyset$ coincide with the finite configurations of $\mathcal{E}$. Again, the proof is straightforward.

Lemma 2.12. *For all* ES *$\mathcal{E}$, and for all $C \subseteq E$:*

$$C \in \mathcal{F}_{\mathcal{E}} \iff \forall D \subseteq_{fin} C.\ \exists C_0.\ D \subseteq C_0 \subseteq_{fin} C.\ \emptyset \rightarrow^{*}_{\mathcal{E}} C_0$$

3. A game-based model of contracts

In this section we present a game-based model for contracts, originally introduced in [12].

3.1. Contracts

A contract (Definition 3.1) specifies the obligations and the objectives of a set of participants, which are ranged over by $\mathsf{A}, \mathsf{B}, \ldots$ in universe $\mathcal{P}_{\mathcal{U}}$. We use $\mathcal{P}, \mathcal{P}', \ldots$ to range over sets of participants.

Obligations are modelled as an event structure; we assume that each event is associated to a participant by a function $\pi : \mathbf{E} \to \mathcal{P}_{\mathcal{U}}$. Intuitively, an enabling $X \vdash e$ models the fact that, if all the events in X have happened, then e is an obligation for participant $\pi(e)$. Such obligation may be discharged only by performing e, or by performing any event in conflict with e. For instance, consider an internal choice between two events a and b. This is modelled by an ES with enablings $\vdash a$, $\vdash b$ and conflict $a \# b$. After the choice (say, of a), the obligation b is discharged. For all $\mathsf{A} \in \mathcal{P}_{\mathcal{U}}$, we write $\mathbf{E}_{\mathsf{A}}$ for the set $\{e \in \mathbf{E} \mid \pi(e) = \mathsf{A}\}$.

Objectives are modelled as a function Φ, which associates each participant A and each trace of events σ to a payoff $\Phi \mathsf{A} \sigma$. We assume a rather coarse notion of payoffs: we only have three possible outcomes which represent, respectively, success (1), failure (-1), and tie (0).

Definition 3.1 (Contract). *A contract $\mathcal{C}$ is a tuple $(\mathcal{E}, \Phi)$, where:*

- *$\mathcal{E} = (E, \#, \vdash, \ell)$ is an event structure,*

- $\Phi : \mathcal{P}_\mathcal{U} \rightharpoonup E^\infty \to \{-1, 0, 1\}$ *associates each participant and trace with a* payoff

and where, for all $X \vdash e$ *in* $\mathcal{E}$, $\Phi(\pi(e))$ *is defined. We say that* $\mathcal{C}$ *is a contract of participants* $\mathcal{P}$ *whenever* $\Phi\mathsf{A}$ *is defined for all* A *in* $\mathcal{P}$.

Note that Φ is a partial function (denoted with the symbol $\rightharpoonup$), hence a contract does not need to define payoffs for all the participants in $\mathcal{P}_\mathcal{U}$: actually, when A advertises her contract, she will not speculate about the objectives of B. The constraint on Φ required by Definition 3.1 asks that if a contract defines some obligations for A, then A must also declare in $\mathcal{C}$ her payoffs.

3.2. Plays

We interpret a contract $\mathcal{C} = (\mathcal{E}, \Phi)$, as a multi-player game [13]. The game involves participants who concurrently perform events in order to reach the objectives defined by Φ. A *play* of the game is a (finite or infinite) trace of the LTS induced by $\mathcal{E}$ according to Definition 2.9.

Definition 3.2 (Play). *A* play *of a contract* $\mathcal{C} = (\mathcal{E}, \Phi)$ *is a (finite or infinite) trace* σ *of* $(\emptyset, \rightarrow_\mathcal{E})$.

Since only enabled events are allowed, as a consequence of Lemma 2.12 we have that plays and configurations share the same events.

Lemma 3.3. *For all plays* σ *of* $(\mathcal{E}, \Phi)$, *the set* $\overline{\sigma}$ *is a configuration of* $\mathcal{E}$.

Each participant can choose a strategy to decide which of her events has to be done at each move. A strategy can only prescribe to perform events enabled by the already occurred ones. When a participant acts as suggested by the strategy, the resulting play is said to *conform* to that strategy.

Definition 3.4 (Strategy and conformance). *A* strategy Σ *for* A *is a function which maps each finite play* $\sigma = \langle e_0 \cdots e_n \rangle$ *to a set of events of* A *(possibly empty), such that*

$$e \in \Sigma(\sigma) \implies \sigma e \text{ is a play}$$

We say that a strategy Σ *is* deterministic *if* $|\Sigma(\sigma)| \leq 1$ *for all plays* σ.
We say that a play $\sigma = \langle e_0 e_1 \cdots \rangle$ conforms *to a strategy* Σ *for* A *if for all* $i \geq 0$,

$$e_i \in \mathbf{E}_\mathsf{A} \implies e_i \in \Sigma(\sigma_i)$$

3.3. Some examples

Example 3.5. *Suppose there are two kids who want to play together. Alice (*A*) has a toy airplane, while Bob (*B*) has a bike. Both kids are willing to share their toys, but they do not trust each other. Thus, before starting to play they advertise the following contracts. Alice will lend her airplane only* after *Bob has allowed her ride his bike. Bob will lend his bike unconditionally. We model the events "Alice lends her airplane" and "Bob lends his bike" as* a *and* b, *respectively. The obligations of Alice and Bob are modelled by the following* ES *(its conflict relations are empty, and the labelling irrelevant):*

$$\mathcal{E}: \quad b \vdash a \ , \ \vdash b$$

The objectives of the two kids are modelled by the function Φ *below. Alice has a positive payoff in those traces where* b *has been performed, while she has a negative payoff when she performs* a *while not obtaining* b *in return. The payoffs of Bob are dual. Formally:*

$$\Phi\mathsf{A} = \lambda\sigma. \begin{cases} 1 & \text{if } b \in \overline{\sigma} \\ 0 & \text{if } a, b \notin \overline{\sigma} \\ -1 & \text{otherwise} \end{cases} \qquad \Phi\mathsf{B} = \lambda\sigma. \begin{cases} 1 & \text{if } a \in \overline{\sigma} \\ 0 & \text{if } b, a \notin \overline{\sigma} \\ -1 & \text{otherwise} \end{cases}$$

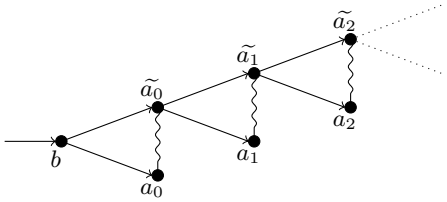


Figure 2: A contract with an indefinitely delayed obligation.

Example 3.6. *Suppose Bob lends his bike to Alice (event b), and requires Alices's toy airplane in exchange. Alice agrees to lend the airplane, but she does not specify the day she will give it. She says she* might *lend it in the same day (event a_0), or one day after (event a_1), or two days after (a_2), and so on. If on day n, Alice is not going to lend the airplane, she fires the event $\widetilde{a_n}$, in conflict with a_n. The obligations of Alice and Bob are modelled by the following* ES*:*

$$\mathcal{E}: \qquad \vdash b,\ \ b \vdash a_0,\ \ b \vdash \widetilde{a_0} \qquad \{\widetilde{a_i} \vdash a_{i+1},\ \widetilde{a_i} \vdash \widetilde{a_{i+1}} \mid i \geq 0\} \qquad \{\widetilde{a_i} \# a_i \mid i \geq 0\}$$

The overall obligations of Alice and Bob are represented in Figure 2, and their payoffs are given by:

$$\Phi \mathsf{A} \sigma = \begin{cases} 1 & \text{if } b \in \overline{\sigma} \\ 0 & \text{if } b \notin \overline{\sigma} \text{ and } \overline{\sigma} \cap S = \emptyset \\ -1 & \text{otherwise} \end{cases} \qquad \Phi \mathsf{B} \sigma = \begin{cases} 1 & \text{if } \overline{\sigma} \cap S \neq \emptyset \\ 0 & \text{if } b \notin \overline{\sigma} \text{ and } \overline{\sigma} \cap S = \emptyset \\ -1 & \text{otherwise} \end{cases}$$

where $S = \{a_i \mid i \geq 0\}$. We will show in 4.11 that Bob does not agree with the contract — as to be expected, since Alice can indefinitely delay lending her airplane. □

3.4. Offer-request payoffs

The definition of payoff functions in Definition 3.1 is quite liberal. Indeed, it also allows for uncomputable functions, which are of little use in doing anything with a contract. Here we shall focus on a particular class of payoff functions, called *Offer-Request*.

Offer-Request payoffs model the situation in which a participant wants something in exchange for a provided service. Each participant A defines a set of pairs (offer,request) $\{(O^i_{\mathsf{A}}, R^i_{\mathsf{A}})\}_i$, where the offers O^i_{A} are sets of events of A, while requests R^i_{A} are sets of events not of A. For A to be successful, whenever A performs some O^i_{A} in a play (in whatever order), then the play must also contain the corresponding R^i_{A}, and at least one of the requests set has to be performed.

Definition 3.7 (Offer-Request payoff). *We say that Φ is an* Offer-Request payoff for A *iff there exists a (possibly infinite) set $\{(O^i, R^i)\}_{i \in I}$ such that for all $i \in I \subseteq \mathbb{N}$, $O^i \subseteq \mathbf{E}_{\mathsf{A}}$, $\emptyset \neq R^i \subseteq \mathbf{E} \setminus \mathbf{E}_{\mathsf{A}}$, and for all σ:*

$$\Phi \mathsf{A} \sigma \ = \ \begin{cases} 1 & \text{if } (\exists i.\ R^i \subseteq \overline{\sigma}) \ \wedge \ (\forall j.\ O^j \subseteq \overline{\sigma} \implies R^j \subseteq \overline{\sigma}) \\ 0 & \text{if } (\forall i.\ R^i \not\subseteq \overline{\sigma} \ \wedge \ O^i \not\subseteq \overline{\sigma}) \\ -1 & \text{otherwise} \end{cases}$$

If all the sets O^i and R^i are finite, we say that Φ is finite (the index set I may still be infinite).

For instance, the payoff functions Φ_{A} and Φ_{B} in Example 3.5 are O-R payoffs for A and B. The offers and the requests of A and B are, respectively $O^0_{\mathsf{A}} = \{a\} = R^0_{\mathsf{B}}$ and, dually, $O^0_{\mathsf{B}} = \{b\} = R^0_{\mathsf{A}}$. Instead, the payoff of B in Example 3.6 is *not* an O-R payoff: indeed, the offer b must be followed by (at least) one of the events a_i.

Some remarks about O-R payoffs follow.

- A play σ has a negative payoff for a participant A if A has already done what she offered ($O^i \subseteq \overline{\sigma}$) and she has not received what she wanted ($R^i \not\subseteq \overline{\sigma}$).
- If A offers nothing for a non-empty set of requests (e.g. $O^0_{\mathsf{A}} = \emptyset$ and $R^0_{\mathsf{A}} \neq \emptyset$), then in the play ε where no events have been performed, A has a negative payoff. Indeed, $O^0_{\mathsf{A}} = \emptyset \subseteq \overline{\varepsilon}$ but $R^0_{\mathsf{A}} \not\subseteq \overline{\varepsilon}$.
- Specifying the same offer set towards differents request sets (for instance $(\{a\},\{b\})$,$(\{a\},\{c\})$) is equivalent to specifying only the single clause $(\{a\},\{b,c\})$, as the plays with positive/negative playoff are the same.

In some Offer-Request payoffs, the requests of participants may mutually depend on their offers. An O-R payoff is *circular* when it is not possible to satisfy the requests of a set of participants without each participant doing some offer. For instance, the payoffs of Alice and Bob in Example 3.5 are circular, because their requests (a and b, respectively) match exactly their offers.

Definition 3.8 (Circular Offer-Request payoff). *Let $\mathcal{P} \subseteq \mathcal{P}_{\mathcal{U}}$ be a set of participants with $|\mathcal{P}| > 1$. We say that an O-R payoff Φ is* circular for $\mathcal{P}$ *whenever:*

$$\forall \mathcal{J} : \mathcal{P} \to I.\ \exists \mathcal{L} : \mathcal{P} \to I.\ \bigcup_{\mathsf{A}\in\mathcal{P}} O^{\mathcal{L}\mathsf{A}}_{\mathsf{A}} \ \subseteq\ \bigcup_{\mathsf{A}\in\mathcal{P}} R^{\mathcal{J}\mathsf{A}}_{\mathsf{A}} \tag{1}$$

Example 3.9. *Consider participants $\mathcal{P} = \{\mathsf{A},\mathsf{B}\}$ with the following O-R payoffs:*

i	O^i_{A}	R^i_{A}
0	$\{a_0\}$	$\{b_0\}$
1	$\{a_0,a_1\}$	$\{b_0,b_1\}$
2	$\{a_0,a_1,a_2\}$	$\{b_0,b_1,b_2\}$

i	O^i_{B}	R^i_{B}
0	$\{b_0\}$	$\{a_0\}$
1	$\{b_1,b_2\}$	$\{a_0,a_1\}$
2	$\{b_0,b_1,b_2\}$	$\{a_0,a_2\}$

There are 3^2 possible choices for the function $\mathcal{J} : \mathcal{P} \to \{0,1,2\}$. For each of these choices, we have that:

$$\{a_0,b_0\} \ \subseteq\ \bigcup_{\mathsf{A}\in\mathcal{P}} R^{\mathcal{J}\mathsf{A}}_{\mathsf{A}}$$

Therefore, we can satisfy (1) by choosing $\mathcal{L} = \{\mathsf{A}\mapsto 0, \mathsf{B}\mapsto 0\}$. By Definition 3.8, we conclude that the payoffs of A and B are circular. Note that in any play where A and B have a positive payoff, there is a prefix of the play where one of the participants has performed all the events in one of his offers, but she has not received the corresponding requests. For instance, for the play $\sigma = \langle a_0\, b_0\rangle$, A has done all her offers in O^0_{A} in the prefix $\langle a_0\rangle$, but there she has not already received R^0_{A}. If we remove the clause $(O^0_{\mathsf{A}}, R^0_{\mathsf{A}})$, then the payoff is no longer circular (take e.g. $\mathcal{J} = \{\mathsf{A}\mapsto 1, \mathsf{B}\mapsto 0\}$). In this case, there exists a contract with a play $\eta = \langle a_0\, b_0\, b_1\rangle$ where both participants have a positive payoff (because $R^1_{\mathsf{A}} \cup R^0_{\mathsf{B}} \subseteq \overline{\eta}$), but there exists no prefix of η where one of the participants has performed all her offers before receiving the corresponding requests. □

Example 3.10 (Dining retailers [14]). *Around a table, n cutlery retailers are about to have dinner. At the center of the table, there is a large dish of food. Despite the food being delicious, the retailers cannot start eating. To dine properly, each retailer needs a complete cutlery set, consisting of n pieces of different kinds. Much to their dismay, each retailer owns a set of n pieces of cutlery, all of the same kind. The retailers start discussing about trading their cutlery, so that they can finally eat.*

We formalise the retailers payoffs as follows. Each retailer A_i initially owns n pieces of kind i. For all $j \neq i$, the event $e_{i,j}$ models A_i giving a piece of cutlery to retailer A_j. Thus, $\mathbf{E}_{\mathsf{A}_i} = \{e_{i,j} \mid j \neq i\}$. Retailer A_i offers $n-1$ pieces of his cutlery of kind i in exchange for $n-1$ pieces of cutlery of the other kinds.

$$O_i = \{e_{i,j} \mid j \neq i\} \qquad\qquad R_i = \{e_{j,i} \mid j \neq i\}$$

By Definition 3.8, the payoff Φ_i of each retailer is a finite O-R circular payoff. Indeed:

$$\bigcup_{i\in 1..n} O_i \ =\ \{e_{i,j} \mid i \neq j\} \ =\ \bigcup_{i\in 1..n} R_i$$ □

4. Agreements

A crucial notion on contracts is that of *agreement*. Intuitively, when Alice 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 dishonest participants. This does not mean that Alice will always reach a positive payoff in all interactions (which is quite unlikely if the others are not cooperating). Rather, we say that Alice agrees on a contract if either she has a positive payoff, or if she can blame someone else. In an actual implementation of a contract-oriented infrastructure, a judge may provide compensations to Alice, or impose a punishment to the participants who have violated the contract. Here, we shall not explicitly model the judge, and we only focus on the agreement property.

4.1. Basic definitions

Recall from Definition 3.2 that we interpret a contract as a multi-player game, where participants concurrently perform events. The plays of this game are the conflict-free sequences of events, with the further requirement that an event e can be fired in a play σ only if e is obliged (i.e., enabled) by the events previously performed in σ. The behaviour of each participant A is specified by a strategy Σ_{A} (Definition 3.4), defining which events of A will be done at each state of a play.

As usual in concurrency, we shall only consider those *fair* plays where an event infinitely often enabled is eventually performed. Indeed, contracts would make little sense in the presence of unfair plays, because an honest participant willing to perform a promised action could be perpetually prevented (by an unfair scheduler, for instance) from keeping her promise.

Technically, we define fairness with respect to the strategy of a participant. A play is fair for a strategy Σ (say, of A) when the other participants cannot prevent A from doing some action persistently chosen by Σ.

Definition 4.1 (Fair play). *We say that a play $\sigma = \langle e_0\, e_1 \cdots \rangle$ is* fair *for Σ iff, for all $i \leq |\sigma|$ and for all e:*

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

Note that, since $\mathbf{E}$ is denumerable, then all ESs admit fair plays (respect to every strategy).

Lemma 4.2. *A play $\sigma = \langle e_0\, e_1 \cdots \rangle$ is* fair *for Σ iff:*

$$\forall i \leq |\sigma|.\ \nexists e.\ \forall j : i \leq j \leq |\sigma|.\ e \in \Sigma(\sigma_j)$$

Proof. For a play $\sigma = \langle e_0\, e_1 \cdots \rangle$ let the predicates $P(e,i)$ and $Q(e,i)$ be defined as:

$$\begin{aligned} P(e,i) &\triangleq \forall j : i \leq j \leq |\sigma|.\ e \in \Sigma(\sigma_j) \\ Q(e,i) &\triangleq \exists h \geq i.\ e_h = e \end{aligned}$$

Then, Definition 4.1 can be rewritten as: $\forall i \leq |\sigma|.\ \forall e.\ P(e,i) \implies Q(e,i)$.

When $Q(e,i)$ is true, there exists $h \geq i$ such that $e_h = e$, hence $\sigma_h\, e = \sigma_{h+1} \xrightarrow{e}\!\!\!\!\!\!/\ $. Thus, by Definition 3.4 it must be $e \notin \Sigma(\sigma_{h+1})$, which implies $P(e,i)$ to be false.

Therefore, $P(e,i)$ implies $Q(e,i)$, which in turn implies $\neg P(e,i)$. From this we conclude that $P(e,i)$ is false, from which the thesis follows:

$$\sigma \text{ is fair} \iff \forall i \leq |\sigma|.\ \forall e.\ \neg P(e,i) \iff \forall i \leq |\sigma|.\ \neg\exists e.\ P(e,i) \qquad \square$$

During a play, a participant is considered *innocent* if she eventually performs all the events which are persistently enabled. Note that if e is an enabled event, then e is no longer enabled if some event in conflict with it is performed.

Definition 4.3 (Innocence). *We say that* A *is* innocent *in* σ *iff:*

$$\forall i \geq 0.\, \forall e \in \mathbf{E}_{\mathsf{A}}.\, \left(\overline{\sigma_i} \xrightarrow{e} \implies \exists j \geq i.\, \overline{\sigma_j} \not\xrightarrow{e}\right)$$

A strategy Σ *for* A *is innocent iff* A *is innocent in all fair plays which conform to* Σ*.*
If A *is not innocent in* σ*, then we say she is* culpable *in* σ*.*

Not all strategies are innocent. For instance, the one which always prescribes A to do nothing is innocent only in case A really has nothing to do. Nevertheless, it is always possible to define a strategy which guarantees A to be innocent in every (fair) play. One such strategy is the *eager strategy*, which prescribes A to do all her enabled events.

Definition 4.4 (Eager strategy). *We define the* eager strategy $\Sigma^{!}_{\mathsf{A}}$ *for* A *as follows:*

$$\Sigma^{!}_{\mathsf{A}}(\sigma) \;=\; \{e \in \mathbf{E}_{\mathsf{A}} \mid \overline{\sigma} \xrightarrow{e} \}$$

We say that a strategy Σ is greater than the strategy Σ', if for all plays σ, we have that $\Sigma'\sigma \subseteq \Sigma\sigma$. The *eager strategy* $\Sigma^{!}_{\mathsf{A}}$ is the greatest strategy for A. Moreover, since $\Sigma^{!}_{\mathsf{A}}$ makes A innocent, we have the following lemma:

Lemma 4.5. $\Sigma^{!}_{\mathsf{A}}$ *is the greatest innocent strategy for* A*.*

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. This is formalised as the function $\mathcal{W}$ in Definition 4.6 below.

Definition 4.6 (Winning play). *Let* $\mathcal{C} = (\mathcal{E}, \Phi)$ *be a contract of participants* $\mathcal{P}$*. We define the function* $\mathcal{W} : \mathcal{P} \to E^{\infty} \to \{1, 0, -1\}$ *as:*

$$\mathcal{W}\mathsf{A}\sigma \;=\; \begin{cases} \Phi\mathsf{A}\sigma & \text{if all participants are innocent in } \sigma \\ -1 & \text{if } \mathsf{A} \text{ is culpable in } \sigma \\ +1 & \text{otherwise} \end{cases}$$

For a participant A *and a play* σ*, we say that* A wins *(resp.* loses*) in* σ *iff* $\mathcal{W}\mathsf{A}\sigma > 0$ *(resp.* $\mathcal{W}\mathsf{A}\sigma < 0$*).*

Note that in the last case of the definition of $\mathcal{W}\mathsf{A}\sigma$, A is innocent but there exists some $\mathsf{B} \neq \mathsf{A}$ culpable in σ.

Definition 4.7 (Winning strategy). *A strategy* Σ *is* winning *(resp.* losing*) for* A *iff* A *wins (resp. loses) in every fair play conforming to* Σ*.*

Whenever A has a strategy Σ which allows her to win in all fair plays conforming to Σ, then she *agrees* on that contract.

Definition 4.8 (Agreement). *A participant* A agrees *on a contract* $\mathcal{C}$ *if and only if* A *has a winning strategy in* $\mathcal{C}$*. A contract* $\mathcal{C}$ *of participants* $\mathcal{P}$ admits an agreement *whenever each* $\mathsf{A} \in \mathcal{P}$ *agrees on* $\mathcal{C}$*.*

Indeed, if A agrees on a contract, then in any interaction regulated by that contract (whatever are the moves of her opponents), she can win by following her strategy.

4.2. Some examples

Example 4.9. *The contract $\mathcal{C}$ of Example 3.5 admits an agreement. The winning strategies for A and B are, respectively:*

$$\Sigma_{\mathsf{A}}(\sigma) = \begin{cases} \{a\} & \text{if } b \in \overline{\sigma} \text{ and } a \notin \overline{\sigma} \\ \emptyset & \text{otherwise} \end{cases} \qquad \Sigma_{\mathsf{B}}(\sigma) = \begin{cases} \{b\} & \text{if } b \notin \overline{\sigma} \\ \emptyset & \text{otherwise} \end{cases}$$

The strategy Σ_{A} prescribes A to do nothing in the empty play, and to do a as long as b has been done. So the fair plays which are conform to Σ_{A} are ε and $\langle b\, a\rangle$. In ε, B is culpable so A wins; in $\langle b\, a\rangle$ the payoff of A is positive and both participants are innocent, so A wins.

The strategy Σ_{B} prescribes B to do b in the empty play and nothing else. So the fair plays conforming to Σ_{B} are $\langle b\rangle$ and $\langle b\, a\rangle$. In $\langle b\rangle$, A is culpable so B wins; in $\langle b\, a\rangle$ the payoff of B is positive and both participants are innocent, so B wins. □

Example 4.10. *The eager strategy $\Sigma^{!}_{\mathsf{A}}$ is not always winning for A. Consider the contract with $\vdash a$, $\vdash b$, $a\#b$, $\mathbf{E}_{\mathsf{A}} = \{a, b\}$, and $\Phi\mathsf{A}\sigma = 1$ iff $a \in \overline{\sigma}$. We have that $\Sigma^{!}_{\mathsf{A}}(\varepsilon) = \{a, b\}$, but A is losing in the fair play $\sigma = \langle b\rangle$. However, A agrees on $\mathcal{C}$, because the strategy $(\lambda\sigma.\,\text{if } \overline{\sigma} \xrightarrow{a} \text{ then } \{a\} \text{ else } \emptyset)$ is winning for A in $\mathcal{C}$.* □

Example 4.11. *Let $\mathcal{C}$ be the contract in Example 3.6. We have that A agrees on $\mathcal{C}$, while B does not. Indeed, a winning strategy for A is the following:*

$$\Sigma_{\mathsf{A}}(\sigma) \;=\; \begin{cases} \{\widetilde{a}_i\} & \text{if } \overline{\sigma} \xrightarrow{\widetilde{a}_i} \\ \emptyset & \text{otherwise} \end{cases}$$

which prescribes A to delay forever the lending of her airplane. The fair plays conforming to this strategy are: ε and the infinite one $\langle b\, \widetilde{a}_0\, \widetilde{a}_1 \widetilde{a}_2 \cdots\rangle$. In the empty play, B is culpable and hence A wins. In the infinite one, A has a positive payoff and both participants are innocent: hence A wins.

On the contrary, B has not a winning strategy: either he will perform b or not. In the empty strategy, where he does not perform b, he is culpabable in the play ε: hence he loses. Otherwise, if he performs b, he cannot make Alice lending her plane. In the play where Alice decides to lend the plane, he wins, but in the infinite play where Alice delays forever, he has a negative payoff and loses.

4.3. Constructions on strategies

The following theorem establishes a relation between deterministic and non-deterministic strategies, by stating that it is always possible to construct a winning deterministic strategy Σ' from a winning non-deterministic Σ. As observed in Example 4.13 below, a naïve construction of Σ' not always produces a winning strategy. The insight of our construction in Theorem 4.12 is that, to define $\Sigma'(\sigma)$, we take the *longest* suffix of σ which has persistently enabled events by Σ, and we enable the *least* of them.

Theorem 4.12. *If A agrees on $\mathcal{C}$, then there exists a deterministic winning strategy for A in $\mathcal{C}$.*

Proof. Let Σ be a (non-deterministic) winning strategy for A in $\mathcal{C}$. Since the universe of events is a denumerable set, we can fix a bijection between it and the set of natural numbers. This induces a (well-) ordering between events such that every event has only finitely many smaller events. We now define a deterministic winning strategy Σ'. For all σ and $j \leq |\sigma|$, let:

$$A(\sigma, j) = \bigcap_{j \leq i \leq |\sigma|} \Sigma(\sigma_i)$$

Then, let the strategy Σ' be defined as follows:

$$\Sigma'(\sigma) = \begin{cases} \emptyset & \text{if } \Sigma(\sigma) = \emptyset \\ \min A(\sigma, j_0) & \text{otherwise, with } j_0 = \min\{j \mid A(\sigma, j) \neq \emptyset\} \end{cases} \tag{2}$$

Note that $\min A(\sigma, j_0)$ denotes the minimum over events, with respect to the ordering mentioned above.

Now, let σ be a fair play conforming to Σ'. Since $\Sigma'(\eta) \subseteq \Sigma(\eta)$ holds for all η, then σ conforms to Σ. We shall prove that σ is fair for Σ. From this and the fact that Σ is winning for A, we will deduce that A wins in σ, which implies the thesis: Σ' is winning for A.

To prove that σ is fair for Σ, we proceed by contradiction. Assume then that σ is *not* fair for Σ. By Lemma 4.2, this amounts to say that there exist some index i and event e such that $e \in \Sigma(\sigma_j)$ for all $j \geq i$. Then, the set $\{j \mid A(\sigma, j) \neq \emptyset\}$ is non-empty: so, let j_0 be its least element, and let $e_0 = \min A(\sigma, j_0)$.

In particular, we have that:

$$e_0 \in \Sigma(\sigma_j) \qquad \text{for all } j \geq j_0 \tag{3}$$

By definition of e_0 we have that, for all $e_1 < e_0$, there exists $h \geq j_0$ such that $e_1 \notin \Sigma(\sigma_h)$ — otherwise e_0 would not be the minimum. Let $\mathit{last}(e_1)$ denote one such index h. Then, let $j_1 = \max\{\mathit{last}(e_1) \mid e_1 < e_0\}$, which is well-defined as there are only finitely many $e_1 < e_0$ (this relies on the assumption that the universe of events is denumerable, as pointed out above). Note that $j_1 \geq j_0$.

Since e_0 is in $\Sigma(\sigma_h)$ for all $h \geq j_0$, while every $e_1 < e_0$ is not in $\Sigma(\sigma_h)$ for some $h \in j_0..j_1$, by definition of Σ' in Equation (2) it must be $\Sigma'(\sigma_j) = \{e_0\}$ for all $j \geq j_1$. Since σ is fair w.r.t. Σ', then e_0 must be in σ. So, there is some index $h \geq j_1 \geq j_0$ such that $e_0 \in \overline{\sigma_h} \stackrel{e_0}{\not\longrightarrow}$, hence $e_0 \notin \Sigma(\sigma_h)$ — which contradicts (3). □

Example 4.13. *Consider the ES with events $\mathbb{N} \cup \{\infty\}$ (for simplicity, assume that they are all in $\mathbf{E}_{\mathsf{A}}$), empty conflict relation, and enabling relation defined by $\vdash \infty$, $\vdash 0$, and $n \vdash n+1$ for all $n \geq 0$. Let $\Phi_{\mathsf{A}}\sigma$ be positive iff $\mathbb{N} \subseteq \sigma$. Note that the eager strategy is winning for A: however, such strategy is non-deterministic, e.g. because $\Sigma(\langle 0, \ldots, n\rangle) = \{n+1, \infty\}$, for all $n \in \mathbb{N}$. Consider now the deterministic strategy Σ' obtained from Σ by removing the event ∞, i.e. $\Sigma'(\sigma) = \Sigma(\sigma) \setminus \{\infty\}$. We have that Σ' is* not *winning: indeed, the infinite play $\eta = \langle 0, 1, \ldots\rangle$ is fair w.r.t. Σ' (and not for Σ), but A is culpable in η, because ∞ is persistenly enabled and never fired.*

We now study how to compose strategies. Note that the naïve definition (i.e. taking the pointwise union) would lead to unwanted results. Indeed, consider the contract $\mathcal{C}$ with enablings $\vdash a$, $\vdash b$, $\{a\} \vdash a'$, $\{b\} \vdash b'$, and conflicts $a\#b'$, $a'\#b$, and where all the events belong to A. Let $\Phi\mathsf{A}\sigma$ be positive if either $a, a' \in \overline{\sigma}$, or $b, b' \in \overline{\sigma}$. Then, the following two strategies are winning for A in $\mathcal{C}$:

$$\Sigma_a(\sigma) = \begin{cases} \{a\} & \text{if } \overline{\sigma} \xrightarrow{a} \\ \{a'\} & \text{if } \overline{\sigma} \xrightarrow{a'} \\ \emptyset & \text{otherwise} \end{cases} \qquad \Sigma_b(\sigma) = \begin{cases} \{b\} & \text{if } \overline{\sigma} \xrightarrow{b} \\ \{b'\} & \text{if } \overline{\sigma} \xrightarrow{b'} \\ \emptyset & \text{otherwise} \end{cases} \tag{4}$$

Their naïve composition $\Sigma = \lambda\sigma.\Sigma_a(\sigma) \cup \Sigma_b(\sigma)$ is *not* winning. Indeed, $\Sigma(a) = \{a', b\}$, and so $\sigma = \langle a\, b\rangle$ is a fair play conforming to Σ, and such that $\Phi\mathsf{A}\sigma \leq 0$. By Definition 4.7, Σ is not winning for A in σ.

Because of the above issue, the definition of strategy composition (Definition 4.14) is slightly more sophisticated. This definition ensures that the composition of a finite set of winning strategies is winning (Lemma 4.15).

Definition 4.14 (Composition of strategies). *For a set of strategies $\mathcal{S}$, we define the strategy $\bigsqcup \mathcal{S}$ as:*

$$(\bigsqcup \mathcal{S})(\sigma) \;=\; \bigcup\{\Sigma(\sigma) \mid \Sigma \in \mathcal{S} \;\wedge\; \sigma \textit{ conforms to } \Sigma\}$$

According to this definition, the composition $\Sigma' = \Sigma_a \sqcup \Sigma_b$ of the strategies in (4) is winning for A:

$$\Sigma'(\sigma) = \begin{cases} \{a, b\} & \text{if } \sigma = \emptyset \\ \{a'\} & \text{if } \sigma = \langle a\rangle \\ \{b'\} & \text{if } \sigma = \langle b\rangle \\ \emptyset & \text{otherwise} \end{cases}$$

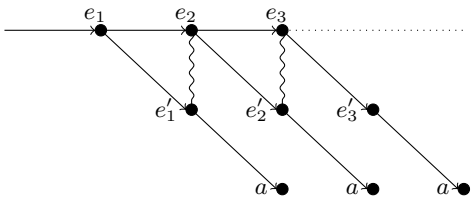


Figure 3: Joining an infinite set of winning strategies is not a winning strategy.

Lemma 4.15. *Let $\mathcal{S}$ be a non-empty finite set of strategies for participant A. Then:*

(a) If a play σ conforms to $\bigsqcup \mathcal{S}$, then there exists $\Sigma \in \mathcal{S}$ such that σ conforms to Σ.

(b) If each $\Sigma \in \mathcal{S}$ is winning (resp. non-losing) for A in $\mathcal{C}$, then $\bigsqcup \mathcal{S}$ is a winning (resp. non-losing) strategy for A in $\mathcal{C}$.

Proof. For item (a), we prove the contrapositive. Assume that σ does not conform to any $\Sigma \in \mathcal{S}$. By Definition 3.4, this means that:

$$\forall \Sigma \in \mathcal{S}.\ \exists i_\Sigma \geq 0.\ \pi(e_{i_\Sigma}) = \mathsf{A} \ \wedge \ e_{i_\Sigma} \notin \Sigma(\sigma_{i_\Sigma}) \tag{5}$$

Clearly, $\sigma_{i_\Sigma+1}$ does not conform to Σ, and so for all $j > i_\Sigma$, σ_j does not as well. Since $\mathcal{S}$ is finite, we can take the maximum of the indices i_Σ obtained in (5) i.e. let:

$$k \ = \ \max \{i_\Sigma \mid \Sigma \in \mathcal{S}\}$$

By construction of k, $\pi(e_k) = \mathsf{A}$, but σ_{k+1} does not conform to any $\Sigma \in \mathcal{S}$. Then, by Definition 4.14, σ does not conform to $\bigsqcup S$.

To prove item (b), let σ be a play conforming to $\bigsqcup \mathcal{S}$. By item (a), there exists $\Sigma \in \mathcal{S}$ such that σ conforms to Σ. Since by hypothesis Σ is winning (resp. non-losing), then A wins in σ. So $\bigsqcup \mathcal{S}$ is a winning (resp. non-losing) strategy for A. □

Note that Lemma 4.15 cannot be applied when the set of strategies is *infinite*. Indeed, for each event e_i of an infinite play σ fair and conforming to $\bigsqcup \mathcal{S}$, there may exists a different $\Sigma_i \in \mathcal{S}$ to whom each σ_i conforms, but not a single Σ to which the *whole* σ conforms. So, even if all the strategies in $\mathcal{S}$ are winning, $\bigsqcup \mathcal{S}$ may not be winning, as shown in the following example.

Example 4.16. *Let $\mathcal{C}_\mathsf{A} = (\mathcal{E}_\mathsf{A}, \Phi_\mathsf{A})$ be a contract with the following payoff:*

$$\Phi \mathsf{A} \sigma \ = \ \begin{cases} 1 & \text{if } a \in \overline{\sigma} \\ -1 & \text{otherwise} \end{cases}$$

where $\mathcal{E}_\mathsf{A}$ has the following enablings and conflicts (see Figure 3):

$$\begin{aligned} \vdash \ &: \ \{\vdash e_1\} \ \cup \ \{e_i \vdash e_{i+1} \mid i \geq 1\} \ \cup \ \{e_i \vdash e'_i \mid i \geq 1\} \ \cup \ \{e'_i \vdash a \mid i \geq 1\} \\ \# \ &: \ \{e'_i \# e_{i+1} \mid i \geq 1\} \end{aligned}$$

and where $\pi(a) = \pi(e_i) = \pi(e'_i) = \mathsf{A}$, for all $i \in \mathbb{N}$.

For all $i > 0$, let Σ^i be the strategy for A which prescribes to wait $i+1$ events before performing a:

$$\Sigma^i(\sigma) \ = \ \begin{cases} \{e_j\} & \text{if } |\sigma| < i \text{ and } \overline{\sigma} \xrightarrow{e_j} \\ \{e'_i\} & \text{if } |\sigma| = i \text{ and } \overline{\sigma} \xrightarrow{e'_i} \\ \{a\} & \text{if } |\sigma| = i+1 \text{ and } \overline{\sigma} \xrightarrow{a} \\ \emptyset & \text{otherwise} \end{cases}$$

The strategy Σ^i is winning for all $i > 0$. Indeed, each fair play σ conforming to Σ^i has the form $\langle e_1 e_2 \cdots e_i e_i' a \rangle$. For instance, $\langle e_1 e_1' a \rangle$ is a fair play conforming to Σ^1 and $\langle e_1 e_2 e_2' a \rangle$ is a fair play conforming to Σ^2. In the play σ, the payoff of A is positive, hence Σ^i is winning for A.

Now, let $\mathcal{S} = \bigsqcup\{\Sigma^i \mid i > 0\}$, and let $\sigma^\infty = \langle e_1\ e_2\ e_3 \ldots \rangle$ be the only infinite play of $\mathcal{C}_\mathsf{A}$. We have that:

- *σ^∞ is fair for $\bigsqcup \mathcal{S}$. Indeed, there does not exist an event e such that $\exists i.\ \forall j \geq i.\ e \in (\bigsqcup \mathcal{S})\sigma^\infty_j$. Thus, Lemma 4.2 states σ^∞ is fair;*
- *σ^∞ conforms to $\bigsqcup \mathcal{S}$, since for all $i > 0$, σ^∞_i conforms to every Σ_j with $j > i$;*
- *A loses in σ^∞, since she never performs the event a.*

Summing up, we have found a fair play conforming to $\bigsqcup \mathcal{S}$ where A is not winning: therefore, $\bigsqcup \mathcal{S}$ is not a winning strategy. □

4.4. Agreements for Offer-Request payoffs

We now provide some general results about contracts with O-R payoffs. The following lemma states a necessary condition to reach an agreement: the ES of the contract must have a configuration containing at least a request set.

Lemma 4.17. *Let $\mathcal{C} = (\mathcal{E}, \Phi)$ be a contract with O-R payoff $\Phi\mathsf{A} = \lambda\sigma.\ \phi(\overline{\sigma})$ for A. If A agrees on $\mathcal{C}$, then there exists some configuration $C \in \mathcal{F}_\mathcal{E}$ such that $\phi(C) > 0$.*

Proof. Assume that A agrees on $\mathcal{C}$. By Definition 4.8, A has a winning strategy in $\mathcal{C}$, be it Σ_A. By Definition 4.7, A wins in every fair play which conforms to Σ_A. Among all these plays, there must exist at least one where all the participants are innocent (e.g. the play where all $\mathsf{B} \neq \mathsf{A}$ adopt the eager strategy $\Sigma^!_\mathsf{B}$), call it σ. Since A wins in σ, by Definition 4.6 we have $\Phi\mathsf{A}\sigma > 0$. To conclude, it suffices to observe that by Lemma 3.3, the set $\overline{\sigma}$ is a configuration of $\mathcal{E}$. □

The following example shows that the converse of Lemma 4.17 does not hold. Indeed, to agree on a contract it is not enough to require that $\phi(C) > 0$ for some $C \in \mathcal{F}_\mathcal{E}$: a conflict may prevent A from reaching a positive payoff.

Example 4.18. *Let $\mathcal{C} = (\mathcal{E}, \Phi)$, where Φ has O-R payoffs for A, be defined as follows:*

$$\begin{array}{lllllll} \mathcal{E}\ : & \vdash a & \vdash a' & \vdash b & \vdash b' & a \# a' & b \# b' \\ \Phi\ : & O^0_\mathsf{A} = \{a\} & R^0_\mathsf{A} = \{b\} & O^1_\mathsf{A} = \{a'\} & R^1_\mathsf{A} = \{b'\} & & \end{array}$$

where $\pi(a) = \pi(a') = \mathsf{A}$, $\pi(b) = \pi(b') = \mathsf{B}$, and the payoff of B is irrelevant. Even though there exist two configurations, $\{a, b\}$ and $\{a', b'\}$, where A has a positive payoff, there are also some plays, e.g. $\langle ab' \rangle$ and $\langle a'b \rangle$, where she has a negative payoff, and hence she loses. Since A has no innocent strategy to avoid these plays, then A does not agree on the contract $\mathcal{C}$. □

The following theorem establishes a sufficient condition for reaching agreements in conflict-free contracts with O-R payoffs. If there exists a configuration C in $\mathcal{C}$ which contains all the requests of A, then A agrees on $\mathcal{C}$. Since the ES of $\mathcal{C}$ is conflict-free, if the strategy of A prescribes to do all her enabled events in C, then the other participants are obliged to do their events in C. Eventually, either some participant $\mathsf{B} \neq \mathsf{A}$ is culpable, or a state is reached where the payoff of A is positive.

Theorem 4.19. *Let $\mathcal{C} = (\mathcal{E}, \Phi)$ be a contract with O-R payoff for A. If $\mathcal{E}$ is conflict-free and $\bigcup_i R^i_\mathsf{A} \subseteq C$ for some configuration $C \in \mathcal{F}_\mathcal{E}$, then A agrees on $\mathcal{C}$.*

Proof. We will prove that the eager strategy $\Sigma^{!}_{\mathsf{A}}$ is winning for A in $\mathcal{C}$. Let γ be a fair play of $\mathcal{C}$ which conforms to $\Sigma^{!}_{\mathsf{A}}$. By contradiction, assume that A is *not* winning in γ. By Lemma 4.5, A is innocent in γ. Thus, by Definition 4.6 it follows that all participants are innocent and, $\Phi\mathsf{A}\gamma \leq 0$. By Definition 3.7, this means that either there exist some i such that $O^i_{\mathsf{A}} \subseteq \overline{\gamma}$ and $R^i_{\mathsf{A}} \not\subseteq \overline{\gamma}$ (in case A loses), or that for all i, $O^i_{\mathsf{A}} \not\subseteq \overline{\gamma}$ and $R^i_{\mathsf{A}} \not\subseteq \overline{\gamma}$ (in case A ties). In both case, there exists at least one i such that $R^i_{\mathsf{A}} \not\subseteq \overline{\gamma}$.

Let i be such that $R^i_{\mathsf{A}} \not\subseteq \overline{\gamma}$, and let e be such that $e \in R^i_{\mathsf{A}} \setminus \overline{\gamma}$. By hypothesis, there exists $C \in \mathcal{F}_{\mathcal{E}}$ such that $\bigcup_i R^i_{\mathsf{A}} \subseteq C$; hence $e \in C$. Since C is a configuration, and since every family of configurations enjoy finiteness, (see [7], Theorem 1.1.9 (ii)) there exists $C' \subseteq_{fin} C$ such that $C' \in \mathcal{F}_{\mathcal{E}}$ and $e \in C'$. By Lemma 2.12, there exists a play $\sigma = \langle e_0 \cdots e_n \rangle$ such that $\emptyset \xrightarrow{\sigma}_{\mathcal{E}} \overline{\sigma}$, and $e \in \overline{\sigma} = C'$.

We will prove that $\overline{\sigma} \subseteq \overline{\gamma}$ by induction on the length of σ. The base case $\sigma_0 = \varepsilon$ is trivial. For the inductive case, we have to prove that $\overline{\sigma_{i+1}} = \overline{\sigma_i} \cup \{e_i\} \subseteq \overline{\gamma}$. By the induction hypothesis, $\overline{\sigma_i} \subseteq \overline{\gamma}$ for $i < |\sigma|$, hence it is enough to prove that $e_i \in \overline{\gamma}$.

Let γ_k be the shortest prefix of γ such that $\overline{\sigma_i} \subseteq \overline{\gamma_k}$. Since $\overline{\sigma_i} \vdash e_i$, by Lemma 2.11 it follows that $\overline{\gamma_h} \xrightarrow{e_i}_{\mathcal{E}}$ for all $h \geq k$. Since all participants are innocent in γ, and since $\mathcal{E}$ is conflict-free, by Definition 4.3 (innocence) there exists some $j > k$ such that the j-th event of γ is e_i — hence $e_i \in \overline{\gamma}$.

Summing up, we have proved that $\bigcup_i R^i_{\mathsf{A}} \subseteq \overline{\gamma}$ for all fair plays γ — contradiction. □

Note that the conflict-freeness requirement in Theorem 4.19 cannot be dropped. Indeed, consider Example 4.18 where we remove $O^1_{\mathsf{A}}, R^1_{\mathsf{A}}$. Then, the configuration $\{b\}$ contains all the requests of A (i.e., R^0_{A}). However, there is no innocent strategy for A where she is winning. Hence, A does not agree on such contract.

5. Session types as contracts

In this section we shall relate the standard progress-based notion of compliance in session types with the notion of agreement in game-based contracts.

Assume that the set of action labels $\mathtt{A}$ is partitioned in two sets, the *output labels* $\mathtt{A}^{!}$, ranged over by $\mathtt{a!}, \mathtt{b!}, \ldots$, and the *input labels* $\mathtt{A}^{?}$, ranged over by $\mathtt{a?}, \mathtt{b?}, \ldots$, We let $\alpha, \beta, \ldots$ range over $\mathtt{A}^{?} \cup \mathtt{A}^{!}$, and we postulate an involution $co(\cdot)$ such that $co(\mathtt{a?}) = \mathtt{a!}$ and $co(\mathtt{a!}) = \mathtt{a?}$. In Section 5.1 we describe the syntax and semantics of *binary* session types, following the notation used in [2]. We then provide session types with an alternative operational semantics, where the two participants alternate in firing actions (Section 5.2); this semantics preserves the progress-based notion of compliance (Theorem 5.11). In Section 5.3 we devise denotational semantics of session types, in the form of an event structure whose LTS is bisimilar to the one of the turn-based operational semantics (Theorem 5.20). Our main result in this section is Theorem 5.23, which states that compliance in session types is equivalent to the winningness of eager strategies in contracts.

5.1. Session types and compliance

Definition 5.1 (Session type). Session types *are terms of the following grammar:*

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

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 terms of a process algebra featuring $\mathbf{1}$ (success), internal choice $\bigoplus_{i\in I} \mathtt{a_i!}\,.\,P_i$, external choice $\sum_{i\in I} \mathtt{a_i?}\,.\,P_i$, and guarded recursion. If $Q = \bigoplus_{i\in I} \mathtt{a_i!}\,.\,P_i$ and $0 \notin I$, we write $\mathtt{a_0!}.P_0 \oplus Q$ for $\bigoplus_{i\in I\cup\{0\}} \mathtt{a_i!}\,.\,P_i$ (same for external choice). As usual, we write $\mathtt{a!}.P$ and $\mathtt{a?}.P$ for singleton choices, we omit trailing occurrences of $\mathbf{1}$, and we assume terms up-to unfolding of recursion.

The semantics of session types is defined in Figure 4. There, we extend the syntax with the term $\mathbf{0}$, which is only used to make it easier relating session types with contracts. 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 $P \mid Q$. An internal choice must first commit to one of the branches $\mathtt{a!}.P$, before advertising $\mathtt{a!}$ (note that, in the first rule

$$\mathtt{a!}.P \oplus Q \to \mathtt{a!}.P \qquad \mathtt{a!}.P \xrightarrow{\mathtt{a!}} P \qquad \mathtt{a?}.P + Q \xrightarrow{\mathtt{a?}} P \qquad \frac{P \to P'}{P \mid Q \to P' \mid Q} \qquad \frac{P \xrightarrow{\mathtt{a!}} P' \quad Q \xrightarrow{\mathtt{a?}} Q'}{P \mid Q \to P' \mid Q'}$$
$$\mathbf{1} \to \mathbf{0}$$

Figure 4: Operational semantics of session types (symmetric rules omitted).

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

Figure 5: Turn-based operational semantics of session types (symmetric rules for B omitted).

of Figure 4, the session type $\mathtt{a!}.P \oplus Q$ has at least two branches). An external choice can always advertise each of its actions. Two session types can run asynchronously only when one of them is committing to a branch. Synchronisation between P and Q requires that P has committed to a branch $\mathtt{a!}$ in an internal choice, and Q is advertising $\mathtt{a?}$ in an external choice.

Following [15, 1, 2] we define a progress-based 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 $P \mid Q$ becomes stuck, the client has reached the success state. Actually, below we provide a more general definition of compliance, which is parametric on the LTS $\to$, parallel composition operator $\circ$, and success states S. The set Z of the states of the LTS $\to$ contains terms of the form $p \circ q$.

Definition 5.2 (Compliance). *Let T be a set of terms, and let $(Z, \circ, \mathtt{A}, \to, S)$ be an LTS, where $Z \subseteq \{p \circ q \mid p, q \in T\}$ and $S \subseteq Z$. We say that p is* compliant *with q (written $p \multimap q$) whenever:*

$$p \circ q \to^* p' \circ q' \not\to \qquad \text{implies} \qquad p' \circ q' \in S$$

Notation 5.3. *Compliance between session types, denoted by $\dashv$, is obtained by instantiating $\to$ in Definition 5.2 with the relation in Figure 4, $\circ$ with parallel $\mid$, and S with the set of terms of the form $\mathbf{0} \parallel Q$.*

Example 5.4. *Let $P = \mathtt{a!} \oplus \mathtt{b!}$, and let $Q = \mathtt{a?}.\mathtt{c!} + \mathtt{d?}$. If P commits to the branch labelled $\mathtt{a}$, then $P \mid Q$ will take a transition to $\mathtt{a!} \mid Q$, which can take a further transition to $\mathbf{1} \mid \mathtt{c!}$. Suppose instead that P commits to $\mathtt{b}$: in this case, $P \mid Q \to \mathtt{b!} \mid Q$, which is stuck because $\mathtt{b}$ is not offered by Q in its external choice. Then, $P \not\dashv Q$ and $Q \not\dashv P$. Instead, with $P' = \mathtt{a!}$ we have that P' is compliant with Q ($P' \dashv Q$), but the* viceversa *is not true ($Q \not\dashv P'$).*

5.2. Turn-based semantics of session types

We now present an alternative operational semantics of session types, where the two terms P and Q in a composition $P \parallel Q$ alternate in firing actions. To do this, we extend the run-time syntax of session types with terms of the form $[\mathtt{a!}]P$, where $[\mathtt{a!}]$ models a one-position buffer storing $\mathtt{a!}$.

Definition 5.5 (Turn-based semantics of session types). *A* turn-based configuration *is a pair $\tilde{P} \parallel \tilde{Q}$, where either: (i) both $\tilde{P}$ and $\tilde{Q}$ are session types, or (ii) $\tilde{P}$ is a session type, and $\tilde{Q} = [\mathtt{a!}]P$ for some session type P (symmetric cases omitted, and including the session type $\mathbf{0}$). In Figure 5 we define an LTS over turn-based configurations, with labels in $\{\mathtt{A}, \mathtt{B}\} \times (\mathtt{A} \cup \{\checkmark\})$.*

A session type with an internal choice $\mathtt{a!}.P \oplus Q$ can fire the action $\mathtt{a!}$ (if the buffer is empty), and write $\mathtt{a!}$ to the buffer. Then, the other session type can read the buffer by firing $\mathtt{a?}$ in an external choice. We also extend the set of labels with $\checkmark$, which is fired by the success state $\mathbf{1}$ before reaching the state $\mathbf{0}$.

The following lemma is straightforward by the rules in Figure 5 and by the assumption (*ii*) in Definition 5.1, which states that actions in an internal/external choice are pairwise distinct.

Lemma 5.6. *The LTS $\xrightarrow{\alpha}\!\!\!\!\rightarrow\!\!\!\!\rightarrow$ is finitely branching. Furthermore, it is deterministic, i.e.:*

$$\tilde{P} \parallel \tilde{Q} \xrightarrow{\alpha}\!\!\!\!\rightarrow\!\!\!\!\rightarrow \tilde{P}_1 \parallel \tilde{Q}_1 \quad \text{and} \quad \tilde{P} \parallel \tilde{Q} \xrightarrow{\alpha}\!\!\!\!\rightarrow\!\!\!\!\rightarrow \tilde{P}_2 \parallel \tilde{Q}_2 \implies \tilde{P}_1 \parallel \tilde{Q}_1 = \tilde{P}_2 \parallel \tilde{Q}_2$$

The notion of compliance under the turn-based semantics is defined by suitably instantiating the parameters in Definition 5.2.

Notation 5.7. *We write $P \dashv\!| Q$ whenever P is compliant with Q, by instantiating $\rightarrow$ in Definition 5.2 with the relation $\twoheadrightarrow$ in Figure 5, $\circ$ with $\|$, and S with the set of turn-based configurations of the form $\mathbf{0} \,\|\, \tilde{Q}$.*

In Theorem 5.11 below we will show that the turn-based compliance of session types is equivalent to the compliance relation of Definition 5.2. To prove that, we introduce a notion of simulation (called *turn-simulation*) which is suitable to relate the turn-based semantics with the one in Figure 4. This relation is between states of two LTSs $\rightarrow_1$ and $\rightarrow_2$, and it is parameterised over two sets S_1 and S_2 of success states. A state $(p_2, \rightarrow_2)$ turn-simulates $(p_1, \rightarrow_1)$ whenever each move of $p1$ can be matched by a sequence of moves of p_2 (ignoring the labels), and stuckness of p_1 implies that p_2 will get stuck in at most one step. Further, turn-simulation must preserve success.

Definition 5.8 (Turn-simulation). *For $i \in \{1, 2\}$, let $\rightarrow_i$ be an LTS over a state space Z_i, and let S_i be a set of states of $\rightarrow_i$. We say that a relation $\mathcal{R} \subseteq Z_1 \times Z_2$ is a* turn-simulation *iff $s_1 \;\mathcal{R}\; s_2$ implies:*

(a) $s_1 \rightarrow_1 s_1' \implies \exists s_2' : s_2 \rightarrow_2^* s_2'$ *and* $s_1' \;\mathcal{R}\; s_2'$

(b) $s_2 \rightarrow_2 s_2' \implies s_1 \rightarrow_1$ *or* $(s_1 \;\mathcal{R}\; s_2'$ *and* $s_2' \not\rightarrow_2)$

(c) $s_2 \in S_2 \implies s_1 \in S_1$

If there is a turn-simulation between s_1 and s_2 (written $s_1 \;\mathcal{R}\; s_2$), we say that s_2 turn-simulates s_1. We denote with $\preccurlyeq$ the greatest turn-simulation.

We say that $\mathcal{R}$ is a turn-bisimulation *iff both $\mathcal{R} \subseteq Z_1 \times Z_2$ and $\mathcal{R}^{-1} \subseteq Z_2 \times Z_1$ are turn-simulations.*

The following lemma relates turn-based simulation and compliance in two arbitrary LTSs. Whenever p and q can be composed in parallel in both LTSs, and these compositions are turn-similar, then compliance can be transferred from one LTS to the other (in the other direction w.r.t. the simulation).

Lemma 5.9. *If $p \circ_1 q \preccurlyeq p \circ_2 q$ and $p \multimap_2 q$, then $p \multimap_1 q$.*

Proof. By Definition 5.2, assume that:

$$p \circ_1 q \;\rightarrow_1^*\; p_1' \circ_1 q_1' \;\not\rightarrow_1$$

Since $p \circ_1 q \preccurlyeq p \circ_2 q$ and $p \circ_1 q \;\rightarrow_1^*\; p_1' \circ_1 q_1'$, by item (a) of Definition 5.8 we have that there exist p_2', q_2' such that $p \circ_2 q \;\rightarrow_2^*\; p_2' \circ_1 q_2'$ and $p_1' \circ_1 q_1' \preccurlyeq p_2' \circ_2 q_2'$. Now we have the following two cases:

- $p_2' \circ_2 q_2' \not\rightarrow_2$. Since $p \multimap_2 q$, by Definition 5.2 it must be $p_2' \circ_2 q_2' \in S_2$. By item (c) of Definition 5.8 it follows that $p_1' \circ_1 q_1' \in S_1$, from which we conclude that $p \multimap_1 q$.
- $p_2' \circ_2 q_2' \rightarrow_2 p_2'' \circ_2 q_2''$. By item (b) of Definition 5.8 we have one of the following two cases:
  - $p_1' \circ_1 q_1' \rightarrow_1$. This is not possible, because we have assumed that $p_1' \circ_1 q_1'$ is stuck.
  - $p_1' \circ_1 q_1' \preccurlyeq p_2'' \circ_2 q_2''$ and $p_2'' \circ_2 q_2'' \not\rightarrow_2$. Since $p_2'' \circ_2 q_2''$ is stuck and $p \multimap_2 q$, by Definition 5.2 it follows that $p_2'' \circ_2 q_2'' \in S_2$. Hence, from $p_1' \circ_1 q_1' \preccurlyeq p_2'' \circ_2 q_2''$ and item (c) of Definition 5.8 it must be $p_1' \circ_1 q_1' \in S_1$, from which the thesis $p \multimap_1 q$ follows. □

The following lemma establishes that the two semantics of session types (Figures 4 and 5) give rise to a turn-bisimulation.

Lemma 5.10. *$P \mid Q$ is turn-bisimilar to $P \,\|\, Q$.*

Proof. Let $(P \mid Q, \to, \{\mathbf{0} \mid \mathbf{0}\})$ be the LTS for $P \mid Q$, and $(P \parallel Q, \twoheadrightarrow, \{\mathbf{0} \parallel \mathbf{0}\})$ be the LTS for $P \parallel Q$. Let $\mathcal{R}$ be a relation defined as follows:

$$\begin{aligned}
\mathcal{R} &= \{(P \mid Q,\ P \parallel Q) \mid P, Q \text{ session types}\} \cup \mathcal{R}_{\mathsf{A}} \cup \mathcal{R}_{\mathsf{B}} \\
\mathcal{R}_{\mathsf{A}} &= \{(\mathtt{a!}.P \mid Q,\ [\mathtt{a!}]P \parallel Q), \quad (\mathtt{a!}.P \mid \mathtt{b!}.Q',\ [\mathtt{a!}]P \parallel \mathtt{b!}.Q' \oplus Q'') \mid P, Q, Q', Q'' \text{ session types}\} \\
\mathcal{R}_{\mathsf{B}} &= \{(P \mid \mathtt{a!}.Q,\ P \parallel [\mathtt{a!}]Q), \quad (\mathtt{a!}.P' \mid \mathtt{b!}.Q,\ \mathtt{a!}.P' \oplus P'' \parallel [\mathtt{b!}]Q) \mid P, P', P'', Q \text{ session types}\}
\end{aligned}$$

Checking that $\mathcal{R}$ is a turn-bisimulation is routine. Full details are on page 32. □

We can now establish that the turn-based compliance of session types is equivalent to the classical notion. This is an immediate consequence of Lemmas 5.9 and 5.10.

Theorem 5.11. *$P \dashv Q$ iff $P \dashv\!\!\mid Q$.*

Proof. Straightforward from Lemma 5.10 and Lemma 5.9

5.3. Denotational semantics of session types

We now provide session types with event structure semantics. We follow two approaches: a *semantical* one, where we encode the LTS of $P \parallel Q$, and a *syntactical* one, where we instead encode P and Q in a syntax-driven fashion, and then combine the resulting event structures. Both approaches have different advantages: the semantical definition is more succinct, while the syntactical one is compositional. In both cases, the denotational semantics gives rise to LTSs on event structures which are bisimilar to the LTS of the turn-based operational semantics of session types in Figure 5. Since the syntactical approach requires quite involved technicalities, to keep our presentation short we establish in this section the correspondence result for the semantical approach only, while developing the other one in Section A.

5.3.1. Semantic-based approach

Given an arbitrary LTS $\mathcal{Z}$, we construct an event structure as follows. The event structure $[\![\mathcal{Z}]\!]$ is crafted so that, whenever the LTS has a trace $\alpha_1\alpha_2\cdots$, the event structure has a trace $(1,\alpha_1)(2,\alpha_2)\cdots$ (according to Definition 2.9). Events of $[\![\mathcal{Z}]\!]$ augment the LTS actions with their index in the trace so that, given a configuration C of $[\![\mathcal{Z}]\!]$, we can reconstruct the original trace.

Definition 5.12 (Encoding of LTSs into ESs). *Let $\mathcal{Z} = (Z, \mathtt{A}, \to, s_0)$ be an LTS with initial state s_0. We define the event structure $[\![\mathcal{Z}]\!] = (E, \#, \vdash, \ell)$ as follows:*

- $E = \{(n, \alpha) \mid n \in \mathbb{N}, \alpha \in \mathtt{A}\}$
- $\# = \{((n,\alpha),(n,\beta)) \mid n \in \mathbb{N}$ *and* $\alpha, \beta \in \mathtt{A}$ *with* $\alpha \neq \beta\}$
- $\vdash = sat(\vdash_{\mathcal{Z}})$, *where* $\vdash_{\mathcal{Z}} = \{(X,(n,\alpha)) \mid s_0 \xrightarrow{snd(X)} s \xrightarrow{\alpha}$ *and* $n = |X| + 1\}$
- $\ell(n,\alpha) = \alpha$

where the partial function snd maps $C = \{(i,\alpha_i)\}_{i\in 1..n}$ to $\langle \alpha_1 \cdots \alpha_n \rangle$.

The finite configurations and the traces of the encoding of an LTS have the expected form, as established by the following lemma.

Lemma 5.13. *Let C be a configuration of $[\![\mathcal{Z}]\!]$. Then:*

(a) if C is finite, then there exist $\alpha_1 \cdots \alpha_{|C|}$ such that $C = \{(i,\alpha_i) \mid i \in 1..|C|\}$.

(b) if $C' \xrightarrow{(n,\alpha)}_{[\![\mathcal{Z}]\!]} C$, then $n = |C'| + 1$ and $C' \vdash_{\mathcal{Z}} (n,\alpha)$.

Proof. For item (a), we proceed by induction on $|C|$. If $|C| = 0$, trivial. Otherwise, by Lemma 2.8 there exists some $e \in C$ such that $C' = C \setminus \{e\} \in \mathcal{F}_{[\![\mathcal{Z}]\!]}$. By the induction hypothesis, there exist α_i such that $C' = \{(1, \alpha_1), \ldots, (|C| - 1, \alpha_{|C|-1})\}$. Let $e = (m, \alpha_m)$. Since C is a configuration of $[\![\mathcal{Z}]\!]$, then $C' \vdash e$. By Definition 5.12, there exists $X \subseteq C'$ such that $X \vdash_{\mathcal{Z}} e$ with $s_0 \xrightarrow{snd(X)} s \xrightarrow{\alpha_m}$ and $m = |X| + 1$. Since C is conflict-free, we must have $m \geq |C|$. Since $X \subseteq C'$, we have $|X| \leq |C'|$, hence $m = |X|+1 \leq |C'|+1 = |C|$. Hence we obtain the thesis $m = |C|$. Note in passing that this implies $X = C'$.

For item (b), by Definition 2.9, we have that $C' \xrightarrow{(n,\alpha)} C$ implies $C = C' \cup \{(n, \alpha)\}$ with $(n, \alpha) \notin C'$, and C' is finite. Applying item (a) to C', C, we have that $C' = \{(i, \alpha'_i) \mid i \in 1..|C'|\}$ and $C = \{(i, \alpha_i) \mid i \in 1..|C|\}$. We then obtain $(n, \alpha) = (|C|, \alpha_{|C|})$, hence $n = |C| = |C'| + 1$. The part of the thesis $C' \vdash_{\mathcal{Z}} (n, \alpha)$ is implied by the proof of item (a), where we established $X = C'$ and $X \vdash_{\mathcal{Z}} e = (n, \alpha)$. □

The following lemma states that the finite traces of the LTS $\mathcal{Z}$ correspond (modulo the projection *snd*) to the finite traces of $[\![\mathcal{Z}]\!]$.

Lemma 5.14. *For all LTSs $\mathcal{Z}$ with initial state s_0:*

(a) $s_0 \xrightarrow{\lambda} s \implies \exists C, \sigma : \lambda = snd(\sigma)$ *and* $\emptyset \xrightarrow{\sigma}_{[\![\mathcal{Z}]\!]} C$

(b) $\emptyset \xrightarrow{\sigma}_{[\![\mathcal{Z}]\!]} C \implies \exists s : s_0 \xrightarrow{\ell(\sigma)} s$

Proof. Item (a) is straightforward by induction on the length of λ. Item (b) follows by Lemma 5.13 and by definition of $\vdash_{\mathcal{Z}}$ in Definition 5.12. □

The following auxiliary result relates the finite traces Tr_{fin} of an LTS with the set Tr of all its traces (including the infinite ones). In particular, we establish that the infinite traces of a finitely-branching LTS are uniquely determined by the finite traces.

Lemma 5.15. *Let $\mathcal{Z}_1, \mathcal{Z}_2$ be finitely-branching LTSs. Then,* $\mathrm{Tr}_{fin}(\mathcal{Z}_1) = \mathrm{Tr}_{fin}(\mathcal{Z}_2) \implies \mathrm{Tr}(\mathcal{Z}_1) = \mathrm{Tr}(\mathcal{Z}_2)$.

Proof. By contradiction, let η be an infinite trace in $\mathcal{Z}_1$ but not in $\mathcal{Z}_2$. Every finite prefix of η is a trace of $\mathcal{Z}_2$. Let $\mathcal{Z}'_2$ be $\mathcal{Z}_2$ restricted to states reachable with any finite prefix of η. The LTS $\mathcal{Z}'_2$ contains traces of arbitrarily large length, and is finitely branching. By König's lemma [16], $\mathcal{Z}'_2$ has an infinite trace η'. The traces η' and η share the same finite prefixes, hence they are equal. □

We also establish finite-branchingness and determinism of our encoding. Recall that the LTS $\rightarrow^{\ell}$ is obtained by substituting actions for events in the labels of an LTS.

Lemma 5.16. *Let $(E, \#, \vdash, \ell) = [\![\mathcal{Z}]\!]$, for some LTS $\mathcal{Z}$. Then, $([\![\mathcal{Z}]\!], \rightarrow^{\ell}_{[\![\mathcal{Z}]\!]})$ is deterministic. Furthermore, if $\mathcal{Z}$ if finitely branching, then $([\![\mathcal{Z}]\!], \rightarrow^{\ell}_{[\![\mathcal{Z}]\!]})$ is finitely branching.*

Proof. For determinism, assume that $C \xrightarrow{\alpha} C'$ and $C \xrightarrow{\alpha} C''$ in the LTS $\rightarrow^{\ell}_{[\![\mathcal{Z}]\!]}$. Then, $C \xrightarrow{(n',\alpha)} C'$ and $C \xrightarrow{(n'',\alpha)} C''$ in $\rightarrow_{[\![\mathcal{Z}]\!]}$. By item (b) of Lemma 5.13, it follows that $n' = n'' = |C| + 1$, and $C' = C'' = C \cup \{(n', \alpha)\}$.

For finite-branchingness, by hypothesis there is a finite number of α such that $s \xrightarrow{\alpha} s'$ in $\mathcal{Z}$. All the configurations of $[\![\mathcal{Z}]\!]$ are reachable from $\emptyset$, hence by item (b) of Lemma 5.14, there is also a finite number of α (and only one $n = |C| + 1$, which is a function of C) such that $C \xrightarrow{(n,\alpha)}$. □

The following theorem relates the denotational and the turn-based operational semantics of session types: their (action-labelled) LTSs are strongly bisimilar.

Theorem 5.17. *For all session types P, Q, we have $(P \parallel Q, \twoheadrightarrow) \sim (\emptyset, \rightarrow^{\ell}_{[\![P \parallel Q]\!]})$.*

Proof. Since $P \parallel Q$ is finitely branching, then by Lemma 5.16 also $(\emptyset, \rightarrow^{\ell}_{[\![P \parallel Q]\!]})$ is finitely branching. By Lemma 5.14, the two LTSs are finite-trace equivalent, and so by Lemma 5.15 they are trace equivalent. Since they are also deterministic (Lemmas 5.6 and 5.16), they are bisimilar. □

$$
\begin{array}{rcl}
[\![\mathbf{1}]\!]^{\mathsf{A}}_{\rho} & = & (\{e\}, \emptyset, sat\{(\emptyset, e)\}, \{(e, \mathsf{A} : \checkmark)\}) \ \text{ where } e \in \mathbf{E}_{\mathsf{A}} \\
[\![x]\!]^{\mathsf{A}}_{\rho} & = & \rho(x) \ = \ (E, \#, \vdash, \ell) \ \text{ where } E \subseteq \mathbf{E}_{\mathsf{A}} \\
[\![\alpha.P]\!]^{\mathsf{A}}_{\rho} & = & (e, \mathsf{A} : \alpha) \centerdot [\![P]\!]^{\mathsf{A}}_{\rho} \ \text{ where } e \in \mathbf{E}_{\mathsf{A}} \ \text{ is a new event} \\
[\![\bigodot_{i \in I} P_i]\!]^{\mathsf{A}}_{\rho} & = & \boxplus_{i \in I} \ [\![P_i]\!]^{\mathsf{A}}_{\rho} \text{ where the } E_i \text{ in } [\![P_i]\!]^{\mathsf{A}}_{\rho} = (E_i, \ \#_i, \ \vdash_i, \ \ell_i) \text{ are pairwise disjoint and } \bigodot \in \{\oplus, +\} \\
[\![rec \ x. \ P]\!]^{\mathsf{A}}_{\rho} & = & fix \ \Gamma \ \text{ where } \Gamma(\mathcal{E}) = [\![P]\!]^{\mathsf{A}}_{\rho\{\mathcal{E}/x\}}
\end{array}
$$

Figure 6: Denotational semantics of session types (operations defined in Section A.3).

$$
(\!|P_1 \parallel P_2|\!)^{\mathsf{A}_1\mathsf{A}_2}_{\rho} = \quad [\![P_1]\!]^{\mathsf{A}_1}_{\rho} \boxdot [\![P_2]\!]^{\mathsf{A}_2}_{\rho} \text{ where } [\![P_i]\!]^{\mathsf{A}_i}_{\rho} = (E_i, \ \#_i, \ \vdash_i, \ \ell_i) \text{ are such that } E_1 \cap E_2 = \emptyset
$$

Figure 7: Denotational semantics of turn-based configurations (full set of rules in Section A.4).

5.3.2. Syntax-based approach

We give here some intuition on the syntactical approach, leaving the most technical details to Section A. To encode a turn-based configuration $P \parallel Q$, we first encode the session types P and Q, and then compose the resulting ESs with the operator $\boxdot$. For configurations having a buffer, such as $[\mathtt{a!}]P \parallel Q$, we proceed in a similar way. Encoding a session type into an ES is almost straightforward (Definition 5.19): events are occurrences of the actions of the session type; all pairs of events belonging to different branches are in conflict; and the guard of a branch enables the events in its continuation.

Definition 5.18 (ES semantics of session types). *The denotation of session types is defined by the rules in Figure 6, where ρ is an environment mapping variables x to* ESs.

The encoding exploits some standard operators to compose event structures, which are defined in Section A.3. Here we simply recall that '$\boxplus$' is the standard choice operator, and '$\centerdot$' is a prefix one. Recursion is dealt with in the usual way, through fixed points; for this, we exploit the complete partial order on event structures in Definition A.16.

The definition of the composition operator $\boxdot$ for a turn-based configuration $\tilde{P} \parallel \tilde{Q}$ is rather involved (see Section A.3), though the intuition behind it is fairly simple. The turn-based interaction of session types depends on their guards and on the one-position buffer: the resulting event structure represents precisely this interaction. For instance, consider the configuration $\mathtt{a!.b!}.\mathbf{1} \parallel \mathtt{a?.b?}.\mathbf{1}$. The ES $\mathcal{E} = [\![\mathtt{a!.b!}.\mathbf{1}]\!]$ contains the enabling $\mathtt{a!} \vdash \mathtt{b!}$, hence to fire it in $\mathcal{E}$ it is enough to fire $\mathtt{a!}$ first. In the encoding of the configuration, the enabling $\mathtt{a!} \vdash \mathtt{b!}$ of $\mathcal{E}$ is enriched by adding the events for the coaction of $\mathtt{a!}$, namely $\mathtt{a?}$. The resulting enabling is $\{\mathtt{a!}, \mathtt{a?}\} \vdash \mathtt{b!}$. The ES $\mathcal{E}' = [\![\mathtt{a?.b?}.\mathbf{1}]\!]$ contains the enabling $\mathtt{a?} \vdash \mathtt{b?}$. In the encoding of the configuration, we enrich this enabling not only with the events corresponding to $\mathtt{a?}$, but also with the one triggering $\mathtt{b?}$ itself, namely $\mathtt{b!}$. The resulting enabling is then $\{\mathtt{a!}, \mathtt{a?}, \mathtt{b!}\} \vdash \mathtt{b?}$. As another example, consider the configuration $[\mathtt{a!}]\mathbf{1} \parallel \mathtt{a?}.\mathbf{1}$. The ES associated to $\mathtt{a?}.\mathbf{1}$ contains the enabling $\vdash \mathtt{a?}$. Since the buffer contains $\mathtt{a!}$, the enabling $\vdash \mathtt{a?}$ is kept as it is (without adding $\mathtt{a!}$ in the premises) in the ES associated to the configuration.

Definition 5.19 (ES semantics of turn-based configuration). *The denotation $(\!|P_1 \parallel P_2|\!)$ of turn-based configurations is defined by the rules in Figure 7, where ρ is an environment mapping variables x to* ESs.

The following theorem relates the denotational and the turn-based operational semantics of session types. Their (action-labelled) LTSs are strongly bisimilar.

Theorem 5.20. *For all session types P, Q, we have $(P \parallel Q, \twoheadrightarrow) \sim (\emptyset, \rightarrow^{\ell}_{(\!|P \parallel Q|\!)})$.*

Proof. See Section A.5.

Example 5.21. *We now illustrate with the help of an example the transformation from session types to* ESs*. Consider two participants* A *and* B*, with session types* $P = \mathtt{a!} \oplus \mathtt{b!}.\mathtt{a!}$ *and* $Q = \mathtt{a?}.\mathtt{b?} + \mathtt{b?}.\mathtt{a?}$*, respectively. According to Definition 5.2, the session type of* A *is compliant with that of* B*, while the converse does not hold. Below we construct the* ESs *associated to* P *and* Q*, and the one associated to the turn-based configuration* $P \parallel Q$*. To ease the reading, we decorate actions in* P *and* Q *with the events they will be associated with in the* ESs*. The events of* A *have odd indexes, whereas those of* B *have even ones.*

$$P = \mathtt{a!}_{e_1}.\mathbf{1}_{e_3} \oplus \mathtt{b!}_{e_5}.\mathtt{a!}_{e_7}.\mathbf{1}_{e_9} \quad \text{and} \quad Q = \mathtt{a?}_{e_2}.\mathtt{b?}_{e_4}.\mathbf{1}_{e_6} + \mathtt{b?}_{e_8}.\mathtt{a?}_{e_{10}}.\mathbf{1}_{e_{12}}$$

By Definition 5.19, we have $[\![P]\!]^{\mathsf{A}}_{\rho} = (E_{\mathsf{A}}, \#_{\mathsf{A}}, \vdash_{\mathsf{A}}, \ell_{\mathsf{A}})$*, where (up-to symmetry and saturation):*

$$E_{\mathsf{A}} = \{e_1, e_3, e_5, e_7, e_9\} \qquad \#_{\mathsf{A}} = \left\{ \begin{array}{c} e_1\#e_5, e_1\#e_7, e_1\#e_9, \\ e_3\#e_5, e_3\#e_7, e_3\#e_9 \end{array} \right\} \qquad \vdash_{\mathsf{A}} = \left\{ \begin{array}{c} \vdash e_1, e_1 \vdash e_3, \\ \vdash e_5, e_5 \vdash e_7, \{e_5, e_7\} \vdash e_9 \end{array} \right\}$$

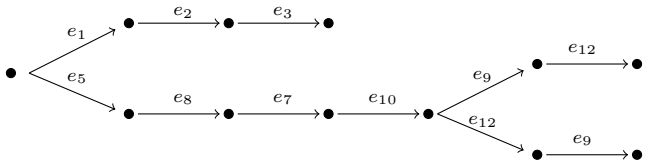
and where $\ell_{\mathsf{A}}(e_1) = \ell_{\mathsf{A}}(e_7) = \mathsf{A} : \mathtt{a!}$*,* $\ell_{\mathsf{A}}(e_5) = \mathsf{A} : \mathtt{b!}$*, and the others are labelled with* $\mathsf{A} : \checkmark$*.*
Similarly, $[\![P]\!]^{\mathsf{B}}_{\rho} = (E_{\mathsf{B}}, \#_{\mathsf{B}}, \vdash_{\mathsf{B}}, \ell_{\mathsf{B}})$*, where:*

$$E_{\mathsf{B}} = (\{e_2, e_4, e_6, e_8, e_{10}, e_{12}\} \quad \#_{\mathsf{B}} = \left\{ \begin{array}{c} e_2\#e_8, e_2\#e_{10}, e_2\#e_{12} \\ e_4\#e_8, e_4\#e_{10}, e_4\#e_{12} \\ e_6\#e_8, e_6\#e_{10}, e_6\#e_{12} \end{array} \right\} \quad \vdash_{\mathsf{B}} = \left\{ \begin{array}{c} \vdash e_2, e_2 \vdash e_4, \{e_2, e_4\} \vdash e_6 \\ \vdash e_8, e_8 \vdash e_{10}, \{e_8, e_{10}\} \vdash e_{12} \end{array} \right\}$$

and where $\ell_{\mathsf{B}}(e_2) = \ell_{\mathsf{B}}(e_{10}) = \mathsf{B} : \mathtt{a?}$*,* $\ell_{\mathsf{B}}(e_4) = \ell_{\mathsf{B}}(e_8) = \mathsf{B} : \mathtt{b?}$*, and the other events are labelled* $\mathsf{B} : \checkmark$*.*
The event structure associated to $P \parallel Q$ *is* $(\!|P \parallel Q|\!)^{\mathsf{A},\mathsf{B}}_{\emptyset} = (E_{\mathsf{A}} \cup E_{\mathsf{B}},\ \#_{\mathsf{A}} \cup \#_{\mathsf{B}},\ \vdash,\ \ell_{\mathsf{A}} \cup \ell_{\mathsf{B}})$*, where:*

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

The event-labelled transition system of $(\!|P \parallel Q|\!)$ *is depicted below:*



5.4. Compliance as agreement

We now exploit the denotational semantics of Section 5.3 to define a transformation from session types P, Q to contracts $\mathcal{C}(P \parallel Q)$. While we will follow the semantical approach, our technical development only relies on the fact that Theorem 5.17 holds; since this is the case also for the syntactical approach (Theorem 5.20), all the results below hold for both the encodings.

Hereafter, we assume that A is the participant running P, while B is running Q. The ES in $\mathcal{C}(P \parallel Q)$ is obtained through Definition 5.12, and the payoff of a participant A is positive in two cases: either a play is infinite, or A has fired the action $\checkmark$.

Definition 5.22 (Contract of a session type). *For all session types* P, Q*, we define the contract* $\mathcal{C}(P \parallel Q)$ *as* $([\![P \parallel Q]\!], \Phi)$*, where:*

$$\Phi A\sigma \ = \ \begin{cases} 1 & \text{if } \sigma \in \mathbf{E}^{\infty} \setminus \mathbf{E}^{*} \text{ or } \exists e \in \overline{\sigma} \cap \mathbf{E}_A : \ell(e) = A : \checkmark \\ -1 & \text{otherwise} \end{cases} \qquad \text{for } A \in \{\mathsf{A}, \mathsf{B}\}$$

We now prove our main result of this section: compliance in session types is equivalent to the winningness of eager strategies in the associated contracts.

Theorem 5.23. *$P \dashv Q$ if and only if the eager strategy is winning for A in $\mathcal{C}(P \parallel Q)$.*

Proof. ($\Rightarrow$) Let $\mathcal{C}(P \parallel Q) = (\mathcal{E}, \Phi)$. By contradiction, assume that $P \dashv Q$, but the eager strategy $\Sigma^!_{\mathsf{A}}$ is not winning for A in $\mathcal{E}$. Since $\Sigma^!_{\mathsf{A}}$ is an innocent strategy (Lemma 4.5), by Definition 4.7 there exists a fair play σ conforming to $\Sigma^!_{\mathsf{A}}$ such that, $\Phi\mathsf{A}\sigma < 1$ and both A and B are innocent. By Theorem 5.17, the turn-based semantics may perform $\ell(\sigma)$, obtaining $P \parallel Q \xrightarrow{\ell(\sigma)}\!\!\!\!\rightarrow$. We have the following three cases:

- σ is an infinite trace. This case does not apply, because by Definition 5.22 A would have a positive payoff.
- $P \parallel Q \xrightarrow{\ell(\sigma)}\!\!\!\!\rightarrow P' \parallel Q'$, and $P' \parallel Q'$ is *not* stuck. This case does not apply since both participants are innocent.
- $P \parallel Q \xrightarrow{\ell(\sigma)}\!\!\!\!\rightarrow P' \parallel Q'$, and $P' \parallel Q'$ is stuck. Since $P \dashv Q$, by Theorem 5.11 we have $P' = \mathbf{0}$. Then, there exists an event $e \in \overline{\sigma}$ such that $\ell(e) = \mathsf{A} : \checkmark$, and so $\Phi\mathsf{A}\sigma = 1$ — contradiction.

($\Leftarrow$) Let $\mathcal{E} = [\![P \parallel Q]\!]$. Assume that the eager strategy $\Sigma^!_{\mathsf{A}}$ is winning for A in $\mathcal{E}$, and that $P \parallel Q \xrightarrow{\nu}\!\!\!\!\rightarrow P' \parallel Q' \not\rightarrow$. We now prove that $P' = \mathbf{0}$. By Theorem 5.17, there exists σ such that $\emptyset \xrightarrow{\sigma} \overline{\sigma} \not\rightarrow$, with $\ell(\sigma) = \nu$, hence both A and B are innocent in σ. Since no event is enabled after σ, σ is fair and conform to the eager strategy $\Sigma^!_{\mathsf{A}}$ in $\mathcal{E}$. Since everyone is innocent and the eager strategy is winning, it must be the case that $\Phi\mathsf{A}\sigma = 1$, and hence there exists $e \in \overline{\sigma}$ such that $\ell(e) = \mathsf{A} : \checkmark$. This allows us to conclude that $P' = \mathbf{0}$, from which we conclude that $P \dashv Q$. □

By the theorem above, it follows that compliance implies agreement, as stated in Corollary 5.24.

Corollary 5.24. *If $P \dashv Q$, then A agrees on $\mathcal{C}(P \parallel Q)$.*

Note that the converse implication does *not* hold, i.e. the fact that A agrees on $\mathcal{C}(P \parallel Q)$ does *not* imply that $P \dashv Q$. For instance, for $P = \mathtt{a!.c!} \oplus \mathtt{b!}$ and $Q = \mathtt{a?} + \mathtt{b?}$, we have that $P \not\dashv Q$, but A agrees on $\mathcal{C}(P \parallel Q)$. Indeed, choosing the branch $\mathtt{b!}$ leads to a winning strategy for A.

Moreover, the converse of Corollary 5.24 does not hold also in case we weaken the hypothesis (i.e. $P \dashv Q$ or $P \vdash Q$) and strengthen the thesis (i.e. $\mathcal{C}(P \parallel Q)$ admits an agreement). The following is a counterexample:

$$P = \mathtt{a!.e?} \oplus \mathtt{b!.(c?} + \mathtt{d?)} \qquad Q = \mathtt{a?} + \mathtt{b?.(c!} \oplus \mathtt{d!.e?)}$$

Indeed, P and Q are not compliant (in either direction), but A can win by choosing the $\mathtt{b!}$-branch, while B can win by choosing the $\mathtt{c!}$-branch.

Example 5.25. *Recall the session types and their associated* ES*s from Example 5.21. We can see that A wins in all the fair plays which conform to the eager strategy $\Sigma^!_{\mathsf{A}}$:*

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

Since $\Sigma^!_{\mathsf{A}}$ is winning, then A agrees on $\mathcal{C}(P \parallel Q)$. Then, by Theorem 5.23, $P \dashv Q$. 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 5.23 it follows that $Q \not\dashv P$.

6. Protection

In contract-oriented interactions, participants advertise their contracts to a contract broker. The broker composes contracts which admit an agreement, and then establishes a session among the participants involved in them [17]. In such scenario, the broker guarantees that — even in the presence of malicious participants — no interaction driven by the contract will ever go wrong. At worst, if some participant does not reach her objectives, then some other participant will be culpable of a contract infringement.

In the above workflow, it is crucial that contract brokers are honest, that is they never establish a session in the absence of an agreement among all the participants. Recall the scenario of Example 3.5, where Alice is willing to lend her airplane in exchange of Bob's bike. In her contract, she could promise to lend the airplane (unconditionally), and declare that her objective is to obtain the bike. A malicious contract broker could construct an attack by establishing a session between Alice and Mallory, whose contract just says to take the airplane and give nothing in exchange. Mallory is *not* culpable, because her contract declares no obligations, and so Alice loses.

To overcome this issue, we study when a contract *protects* a participant from dishonest brokers. Formally, a contract $\mathcal{C}_{\mathsf{A}}$ protects A if, whatever contract $\mathcal{C}$ is *composed* by the broker with $\mathcal{C}_{\mathsf{A}}$, A has a way to either win or tie in the composed contract. We start by formalising contract composition.

6.1. Contract composition

Given two contracts $\mathcal{C}$ and $\mathcal{C}'$, we denote with $\mathcal{C} \mid \mathcal{C}'$ their composition. This is a partial operation: $\mathcal{C}$ and $\mathcal{C}'$ are composable only if they are not defining payoffs for the same participant. Also, the contracts must agree on the labelling of events.

Definition 6.1 (Composition of contracts). *Let $\mathcal{E} = (E, \vdash, \#, \ell)$, and let $\mathcal{E}' = (E', \vdash', \#', \ell')$. Two contracts $\mathcal{C} = (\mathcal{E}, \Phi)$ and $\mathcal{C}' = (\mathcal{E}', \Phi')$ are* composable *whenever:*

$$\forall \mathsf{A} \in \mathcal{P}_{\mathcal{U}}.\ \Phi(\mathsf{A}) = \bot \ \vee \ \Phi'(\mathsf{A}) = \bot \tag{6}$$

$$\forall e \in E \cap E'.\ \ell(e) = \ell'(e) \tag{7}$$

If $\mathcal{C}$, $\mathcal{C}'$ are composable, we define their composition $\mathcal{C} \mid \mathcal{C}'$ as $(\mathcal{E} \sqcup \mathcal{E}', \Phi \cup \Phi')$.

The following lemma states that two contracts which both assign obligations to A are not composable.

Lemma 6.2. *If $\mathcal{C} = (\mathcal{E}, \Phi)$ and $\mathcal{C}' = (\mathcal{E}', \Phi')$ are composable, then for all e, e', X, X', we have:*

$$X \vdash e \in \mathcal{E} \ \wedge \ X' \vdash e' \in \mathcal{E}' \implies \pi(e) \neq \pi(e')$$

Proof. Let $X \vdash e \in \mathcal{E}$ and $X' \vdash e' \in \mathcal{E}'$. By contradiction, let us assume that $\pi(e) = \pi(e') = \mathsf{A}$. By Definition 3.1 we have that $\Phi\mathsf{A} \neq \bot$ and $\Phi'\mathsf{A} \neq \bot$, which contradicts condition (6) in Definition 6.1. □

6.2. Definition of protection

Definition 6.3 (Protection). *A contract $\mathcal{C}_{\mathsf{A}}$* protects *participant A whenever, for all contracts $\mathcal{C}$ composable with $\mathcal{C}_{\mathsf{A}}$, A has a non-losing strategy in $\mathcal{C}_{\mathsf{A}} \mid \mathcal{C}$.*

Note that if A agrees with $\mathcal{C}$, then not necessarily $\mathcal{C}$ protects A. For instance, Mallory could join $\mathcal{C}$ with her contract $\mathcal{C}_{\mathsf{M}}$, and prevent Alice from borrowing Bob's bike in $\mathcal{C} \mid \mathcal{C}_{\mathsf{M}}$. A sufficient (yet hardly realistic) criterion for protection would be to declare nonnegative payoffs for all σ. Less trivially, the following example shows a contract with possible negative payoffs which still offers protection.

Example 6.4. *Recall the contract $\mathcal{C}$ in Example 3.5. This can be obtained by composing Alice's contract $\mathcal{C}_{\mathsf{A}}$ and Bob's contract $\mathcal{C}_{\mathsf{B}}$, defined in the natural way. The contract $\mathcal{C}_{\mathsf{B}}$ does* not *protect Bob. To prove that, consider e.g. the attacker contract $\mathcal{C}' = (\mathcal{E}', \Phi_{\mathcal{C}'})$, where we define $\mathcal{E}'$ with no enablings, and $\Phi_{\mathcal{C}'}$ is not*

relevant except for being undefined on B *(otherwise* $\mathcal{C}'$ *and* $\mathcal{C}_\mathsf{B}$ *would not be composable). Consider then the contract* $\mathcal{C}' \mid \mathcal{C}_\mathsf{B}$*. There are only two possible strategies for* B*:*

$$\Sigma_\mathsf{B} = \lambda\sigma.\,\emptyset \qquad\qquad \Sigma'_\mathsf{B} = \lambda\sigma.\begin{cases}\{b\} & \text{if } b \notin \overline{\sigma} \\ \emptyset & \text{otherwise}\end{cases}$$

The strategy Σ_B *is losing for* B*, because* B *is not innocent under* Σ_B*. The strategy* Σ'_B *is losing as well, because in the play* $\sigma = \langle b \rangle$ *(fair and conform to* Σ_B*), no participant is culpable (according to* $\mathcal{C}' \mid \mathcal{C}_\mathsf{B}$*) and* $\Phi\mathsf{B}\sigma = -1$*. Hence by Definition 6.3,* B *is not protected by* $\mathcal{C}_\mathsf{B}$*.*

Instead, the contract $\mathcal{C}_\mathsf{A}$ *protects Alice. To show that, consider a contract* $\mathcal{C}$ *composable with* $\mathcal{C}_\mathsf{A}$*. Let* Σ_A *be the following strategy for* A*:*

$$\Sigma_\mathsf{A} = \lambda\sigma.\begin{cases}\{a\} & \text{if } b \in \overline{\sigma} \text{ and } a \notin \overline{\sigma} \\ \emptyset & \text{otherwise}\end{cases}$$

Let σ *be a play in* $\mathcal{C} \mid \mathcal{C}_\mathsf{A}$ *fair and conform to* Σ_A*. There are two cases:*

- $b \in \overline{\sigma}$*. Since* σ *is fair for* Σ_A*, then either* $a \in \overline{\sigma}$*, or there exists some* $e \in \overline{\sigma}$ *such that* $e \# a$*. In both cases,* A *is innocent in* σ*. Furthermore,* $\Phi\mathsf{A}\sigma = 1$*.*
- $b \notin \overline{\sigma}$*. By definition of* $\mathcal{C}_\mathsf{A}$*, and since* $\mathcal{C}$ *is not specifying any further obligations for* A *(otherwise it would not be composable with* $\mathcal{C}_\mathsf{A}$*), then* A *is not culpable in* σ*. Also, since* $b \notin \overline{\sigma}$ *and* $a \notin \overline{\sigma}$*, then* $\Phi\mathsf{A}\sigma = 0$*.*

In both cases, Σ_A *is non-losing for* A*. Therefore,* $\mathcal{C}_\mathsf{A}$ *protects* A*.* □

6.3. Protection for Offer-Request payoffs

We now study protection in contracts with Offer-Request payoffs. A necessary condition to being protected is to specify non-empty offers sets. In fact if A were specifing an empty set of offers, she would lose in an empty play. Intuitively, A is saying that she wants something by doing nothing in exchange. This means that when *nothing is done*, A expects her requests to be satisfied. So even in the case of an empty set of obligations, A is protected only if she specifies non-empty offer sets.

Example 6.5. *Assume that* $\mathcal{C}_\mathsf{A}$ *has an empty offer associated with a non-empty request:*

$$O^0_\mathsf{A} = \emptyset \qquad R^0_\mathsf{A} \neq \emptyset$$

In case the contract of B *prescribes no obligations for* B*, then* B *is innocent and* A *loses in all plays where no events are performed. Hence* $\mathcal{C}_\mathsf{A}$ *does not protect* A*, as correctly predicted by Lemma 6.6 below.* □

Lemma 6.6. *If the contract* $\mathcal{C}_\mathsf{A} = (\mathcal{E}, \Phi)$ *with O-R payoffs for* A *protects* A*, then* $\forall h.\ O^h_\mathsf{A} \neq \emptyset$

Proof. By Definition 6.3, for every contract $\mathcal{C}$ composable with $\mathcal{C}_\mathsf{A}$, A has a non-losing strategy Σ in $\mathcal{C}_\mathsf{A} \mid \mathcal{C}$. Let $\mathcal{C}$ have no enabling for any of the events in R^i for all i, and let σ be a fair play of $\mathcal{C}_\mathsf{A} \mid \mathcal{C}$ conform to Σ. Since $\mathcal{C}$ has no enablings for any R^i, there exist no h such that $R^h_\mathsf{A} \subseteq \overline{\sigma}$. According to Definition 3.7, the only way for A to lose is to have $O^i_\mathsf{A} \subseteq \overline{\sigma}$ and $R^i_\mathsf{A} \not\subseteq \overline{\sigma}$ for some i. So, since A does not lose in σ, then for all i, $O^i_\mathsf{A} \not\subseteq \overline{\sigma}$, and we conclude that $O^i_\mathsf{A} \neq \emptyset$ for all i. □

A sufficient condition for A to be protected is given in Theorem 6.7: A can promise to do what she offers in the O-R contract, only *after* the other participants have fulfilled her requests. More precisely, A is protected if, whenever she enables an offer O^i_A, the corresponding request R^i_A has already been satisfied.

Theorem 6.7. *A contract* $\mathcal{C}_\mathsf{A} = (\mathcal{E}, \Phi)$ *with O-R payoffs for* A *protects* A *if*

$$\forall i.\ \exists e \in O^i_\mathsf{A}.\ (\forall Y.\ Y \vdash e \implies R^i_\mathsf{A} \subseteq Y) \tag{8}$$

Proof. Let Φ be an O-R payoff for A such that (8) holds. Let $\mathcal{C}$ be a contract composable with $\mathcal{C}_\mathsf{A}$. We will prove that the eager strategy $\Sigma^!_\mathsf{A}$ is non-losing for A in $\mathcal{C}_\mathsf{A} \mid \mathcal{C}$. Let σ be a fair play of $\mathcal{C}_\mathsf{A} \mid \mathcal{C}$ (of course, σ conforms to $\Sigma^!_\mathsf{A}$). By contradiction, assume that A loses in σ, i.e. by Definition 4.6 and by Definition 3.7:

$$\exists i.\ O^i_\mathsf{A} \subseteq \overline{\sigma} \ \wedge\ R^i_\mathsf{A} \not\subseteq \overline{\sigma} \tag{9}$$

For the index i given by (9), let $e \in O^i_\mathsf{A}$ be the event whose existence is given by (8). Since $O^i_\mathsf{A} \subseteq \overline{\sigma}$, then it must be $\overline{\sigma} \vdash e$. By (8) it follows that $R^i_\mathsf{A} \subseteq \overline{\sigma}$ — contradicting (9). □

Example 6.8. *Condition* (8) *in Theorem 6.7 is not necessary for protection. Indeed, in a contract for* A *with no obligations and non-empty offers,* A *would be protected, since she could do nothing and non-lose. Also, if* A *offers an unreachable event,* A *is protected since she will never obliged to do what she offers.* □

6.4. Agreement and protection cannot coexist

A remarkable feature of *finite* circular payoffs is that, in each play where all participants win, at some point there exists a participant A which has performed all the offers in O^i_A before having obtained all the requests in R^i_A. Intuitively, the participant A which makes this "first step" is not protected. The proof technique exploited by Lemma 6.9 is somehow similar to that used in [18] to prove that fair exchange is impossible without a trusted third party.

Lemma 6.9. *Let* $\mathcal{C}$ *be a contract of participants* $\mathcal{P}$*, with finite circular O-R payoffs of* $\mathsf{A}_1, \ldots, \mathsf{A}_n \in \mathcal{P}$*. If* σ *is a winning play for* $\mathcal{P}$*, then there exists some finite prefix* η *of* σ *and some* $k \in 1..n$ *such that* $\Phi \mathsf{A}_k \eta < 0$*.*

Proof. Since σ is a winning play for $\mathcal{P}$, then by Definition 4.6 no participant is culpable in σ, and so it must be $\Phi \mathsf{A} \sigma > 0$ for all $\mathsf{A} \in \mathcal{P}$. By Definition 3.7, σ contains at least a request for all participants in $\mathcal{P}' = \{\mathsf{A}_1, \ldots, \mathsf{A}_n\}$. Since each request set is finite, then there exists a finite prefix of σ which contains (at least) a request of each participant in $\mathcal{P}'$. Let η' be the shortest of such prefixes. Since request sets are non-empty, it must be $\eta' = \eta\, e$, for some η and e.

By the choice of η', there exists some participant $\mathsf{B} = \mathsf{A}_k$ and some request $R^{i_\mathsf{B}}_\mathsf{B}$ such that:

$$e \in R^{i_\mathsf{B}}_\mathsf{B} \ \subseteq\ \overline{\eta} \cup \{e\} \tag{10}$$

$$\forall j : R^j_\mathsf{B} \ \not\subseteq\ \overline{\eta} \tag{11}$$

otherwise η' is not the *shortest* prefix containing a request of each participant. Again, by the choice of η' for all $\mathsf{A} \in \mathcal{P}' \setminus \{\mathsf{B}\}$ we can take i_A such that $R^{i_\mathsf{A}}_\mathsf{A} \subseteq \overline{\eta} \cup \{e\}$. Since Φ is circular, by (1) in Definition 3.8 there exists a function $\mathcal{J} : \mathcal{P}' \to \mathbb{N}$ such that $\overline{\eta} \cup \{e\} \supseteq \bigcup_{\mathsf{A} \in \mathcal{P}'} R^{i_\mathsf{A}}_\mathsf{A} \supseteq \bigcup_{\mathsf{A} \in \mathcal{P}'} O^{\mathcal{J}\mathsf{A}}_\mathsf{A}$. Therefore, $O^{\mathcal{J}\mathsf{B}}_\mathsf{B} \subseteq \overline{\eta} \cup \{e\}$. By Definition 3.7 and by (10), we have that $e \notin \mathbf{E}_\mathsf{B}$, and this in turn implies that $e \notin O^{\mathcal{J}\mathsf{B}}_\mathsf{B}$. From this and $O^{\mathcal{J}\mathsf{B}}_\mathsf{B} \subseteq \overline{\eta} \cup \{e\}$ it follows that $O^{\mathcal{J}\mathsf{B}}_\mathsf{B} \subseteq \overline{\eta}$. Since by (11) we have $R^{\mathcal{J}\mathsf{B}}_\mathsf{B} \not\subseteq \overline{\eta}$, then by Definition 3.7 we conclude that $\Phi \mathsf{B} \eta < 0$. □

Lemma 6.9 does not hold if the payoff is non-circular, as illustrated by the following example.

Example 6.10. *Consider the following (non-circular) O-R payoff for* A*,* B*,* C*:*

$$O^1_\mathsf{A} = \{a, a', a''\} \qquad O^1_\mathsf{B} = \{b\} \qquad O^1_\mathsf{C} = \{c\}$$
$$R^1_\mathsf{A} = \{b, c\} \qquad R^1_\mathsf{B} = \{a, a'\} \qquad R^1_\mathsf{C} = \{b\}$$

In a play $\sigma = \langle a\, a'\, b\, c \rangle$ *every participant is winning, but no one is losing any prefix of* σ*. In particular:*

- *in* $\sigma_2 = \langle a\, a' \rangle$ *(and its prefixes) no participant has done all her offers.*
- *in* $\sigma_3 = \langle a\, a'\, b \rangle$*,* A *and* C *have not done all her offers, and* B *has obtained his requests.* □

The main result of this section follows. Assume that in a contract there is a set participants with finite circular O-R payoffs. The theorem states that if the contract admits an agreement, then some of these participants is not protected.

Theorem 6.11. *Let $\tilde{\mathcal{C}}$ be a contract, let $\mathcal{C}_i$ be a contract of A_i, for all $i \in 1..n$, and let $\mathcal{C} = \mathcal{C}_1 \mid \cdots \mid \mathcal{C}_n \mid \tilde{\mathcal{C}}$ have finite circular O-R payoffs for $\mathsf{A}_1, \ldots, \mathsf{A}_n$. Then, at most one of the following statements is true:*

(a) $\mathcal{C}$ admits an agreement;

(b) for all $i \in 1..n$, $\mathcal{C}_i$ protects A_i.

Proof. Assume that the statement (a) is true, i.e. all the participants agree on $\mathcal{C} = \mathcal{C}_1 \mid \cdots \mid \mathcal{C}_n \mid \tilde{\mathcal{C}}$. By Definition 4.8, each $\mathsf{A}_i \in \{\mathsf{A}_1, \ldots, \mathsf{A}_n\}$ has a winning strategy Σ_i in $\mathcal{C}$. Let σ be a fair play of $\mathcal{C}$ conforming to all the Σ_i. Since all the participants win in σ, by Lemma 6.9 there exists some $k \in 1..n$ and some finite prefix η of σ such that $\Phi \mathsf{A}_k \eta < 0$. By Definition 3.7, this amounts to say that there exists some h such that $O_k^h \subseteq \overline{\eta}$ and $R_k^h \not\subseteq \overline{\eta}$.

We now prove that $\mathcal{C}_k$ does *not* protect A_k. To do that, we construct a contract $\mathcal{C}' = (\mathcal{E}', \Phi')$ such that A_k does not have a non-losing strategy in $\mathcal{C}_k \mid \mathcal{C}'$. The function Φ' in $\mathcal{C}'$ is almost immaterial: we just require that it makes $\mathcal{C}'$ composable with $\mathcal{C}_k$. We define the ES $\mathcal{E}'$ so that it comprises some event $\widetilde{e}$ not occurring in $\mathcal{C}$, and such that its enablings and conflicts are the following:

$$\{ \vdash e \mid e \in \overline{\eta} \setminus \mathbf{E}_{\mathsf{A}_k} \} \cup \{ \vdash \widetilde{e} \}$$
$$\{ e \# \widetilde{e} \mid e \in \mathbf{E}_{\mathsf{A}_k} \setminus \overline{\eta} \}$$

Intuitively, $\mathcal{C}'$ enables all the events in η of each participant different from A_k, and also the event $\widetilde{e}$, which is in conflict with all the events of A_k, *except* for the events in η. The goal of $\mathcal{C}'$ is to force A_k to perform the events in $\overline{\eta}$, which results in O_k^h being done before R_k^h is reached. To implement this goal, the participants of $\mathcal{C}'$ must also be innocent in η.

By contradiction, assume that Σ is a non-losing strategy for A_k in $\mathcal{C}_k \mid \mathcal{C}'$. Assume that all the participants in $\mathcal{C}'$ adopt the eager strategy, i.e. all the enabled events are in their strategy. Then, there exists a fair play $\nu = \langle e_0 e_1 \cdots \rangle$ of $\mathcal{C}_k \mid \mathcal{C}'$ which (a) conforms to Σ and the eager strategies of $\mathcal{C}'$, and where (b) the first event in ν is $\widetilde{e}$. Consequently, (c) all the participants are innocent in ν (because the eager strategy is innocent, by Lemma 4.5). By construction of $\mathcal{C}'$ and by (b), we have $\overline{\nu} \subseteq \overline{\eta} \cup \{\widetilde{e}\}$; hence, ν is a finite play $\langle e_0 \cdots e_m \rangle$. Further, whenever $\overline{\nu_i} \xrightarrow{e}$ (for $i > 0$), then $e \in \overline{\eta}$. By (c) and by Definition 4.3:

$$\forall i > 0.\, \forall e.\, \big(\overline{\nu_i} \xrightarrow{e} \implies \exists j \geq i.\, e_j \# e \ \vee \ e_j = e\big) \tag{12}$$

Since $\overline{\nu_i} \xrightarrow{e}$ implies $e \in \overline{\eta}$, the case $e_j \# e$ in (12) is not possible. We can then rewrite equation (12) as follows:

$$\forall i > 0.\, \forall e.\, \big(\overline{\nu_i} \xrightarrow{e} \implies \exists j \geq i.\, e_j = e\big) \tag{13}$$

We now prove that $\overline{\eta} \subseteq \overline{\nu}$. Let $\eta = \langle e'_1 \cdots e'_{m'} \rangle$. By induction on i, we prove that $\overline{\eta_i} \subseteq \overline{\nu}$. The base case $i = 0$ is trivial. For the inductive case, by the induction hypothesis we have that $\overline{\eta_i} \subseteq \overline{\nu}$, so it remains to prove that $e'_i \in \overline{\nu}$. We have the following two cases:

1. $e'_i \in \mathbf{E}_{\mathsf{A}_k}$. By definition of play, we have that $\overline{\eta_i} \vdash e'_i$ in $\mathcal{C}_k$. Since $\overline{\eta_i} \subseteq \overline{\nu}$ and $CF(\overline{\nu})$, then by saturation we also have $\overline{\nu} \vdash e'_i$. Therefore, there exists some $0 < j \leq m$ such that $\overline{\nu_j} \xrightarrow{e'_i}$. If $j = 0$, then $\vdash e'_i$, and so also $\overline{\nu_1} \xrightarrow{e'_i}$. By (13), we conclude $e'_i \in \overline{\nu}$. If $j > 0$, we exploit directly (13) to infer $e'_i \in \overline{\nu}$.

2. $e'_i \notin \mathbf{E}_{\mathsf{A}_k}$. By definition of $\mathcal{C}'$, we have that $\vdash e'_i$, and so $\overline{\nu_1} \xrightarrow{e'_i}$. By (13), we conclude that $e'_i \in \overline{\nu}$.

Summing up, there exists a fair play ν of $\mathcal{C}_k \mid \mathcal{C}'$ which conforms to Σ, and where $O_k^h \subseteq \overline{\eta} \subseteq \overline{\nu}$. Furthermore, $R_k^h \not\subseteq \overline{\nu}$, because the ES $\mathcal{E}'$ only enables the events in $\overline{\eta}$ (plus the event $\widetilde{e} \notin R_k^h$), while $\overline{\eta}$ does not contain all the events in R_k^h. By Definition 3.7, $\mathcal{W}\mathsf{A}_k \nu = \Phi \mathsf{A}_k \nu < 0$, i.e. A_k loses in ν — contradiction. □

Example 6.12. *Consider the contract* $\mathcal{C}_\mathsf{A}$ *with enabling* $b \vdash a$ *and the finite circular O-R payoff* $O^0_\mathsf{A} = \{a\}$, $R^0_\mathsf{A} = \{b\}$, *and the contract* $\mathcal{C}_\mathsf{B}$ *with enabling* $a \vdash b$ *and payoff* $O^0_\mathsf{B} = \{b\}$, $R^0_\mathsf{B} = \{a\}$.

Every participant is protected by her own contract, but the composed contract $\mathcal{C}_\mathsf{A} \mid \mathcal{C}_\mathsf{B}$ *does not admit an agreement, as correctly predicted by Theorem 6.11.* □

Agreement and protection can coexist in contracts with *infinite* circular O-R payoffs, as shown by the following example. Intuitively, when an infinite offer O_A has to match an infinite request R_B, participants A and B may take turns in doing event in $O_\mathsf{A} \cup R_\mathsf{B}$. This strategy is winning for both participants (hence they have an agreement), and protection follows because no participant completes her offer before receiving the corresponding request.

Example 6.13. *Let* $\mathcal{C}_\mathsf{A} = (\mathcal{E}_\mathsf{A}, \Phi_\mathsf{A})$ *and* $\mathcal{C}_\mathsf{B} = (\mathcal{E}_\mathsf{B}, \Phi_\mathsf{B})$ *be contracts with circular O-R payoffs (with infinite offers/requests) defined as follows:*

$$O_\mathsf{A} \;=\; \{e_i \mid i \in \mathbb{N}\} \;=\; R_\mathsf{B} \qquad\qquad R_\mathsf{A} \;=\; \{\overline{e_i} \mid i \in \mathbb{N}\} \;=\; O_\mathsf{B}$$

and let $\mathcal{P} = \{\mathsf{A}, \mathsf{B}\}$, $\pi(e_i) = \mathsf{A}$, $\pi(\overline{e_i}) = \mathsf{B}$ *for all* $i \in \mathbb{N}$. *Let the* ES $\mathcal{E}_\mathsf{A}$ *and* $\mathcal{E}_\mathsf{B}$ *be defined by the following enablings (and no conflicts):*

$$\mathcal{E}_\mathsf{A}: \; \{\vdash e_0\} \cup \{\overline{e_i} \vdash e_{i+1} \mid i \geq 0\} \qquad\qquad \mathcal{E}_\mathsf{B}: \; \{e_i \vdash \overline{e_i} \mid i \geq 0\}$$

The contract $\mathcal{C} = \mathcal{C}_\mathsf{A} \mid \mathcal{C}_\mathsf{B}$ *admits an agreement. We prove separately that* A *and* B *agree on* $\mathcal{C}$. *Let* $\Sigma^!_\mathsf{A}$ *be the eager strategy for* A. *Let* σ *be a fair play of* $\mathcal{C}$ *conform to* $\Sigma^!_\mathsf{A}$. *We prove that* A *wins in* σ. *By Lemma 4.5, the strategy* $\Sigma^!_\mathsf{A}$ *makes* A *innocent in* σ. *There are two subcases. If* B *is not innocent in* σ, *then* A *wins. Otherwise, the play* σ *must be infinite, i.e.* $\overline{\sigma} = \{e_i\}_{i\in\mathbb{N}} \cup \{\overline{e_i}\}_{i\in\mathbb{N}}$. *Therefore,* $R_\mathsf{A} \subseteq \overline{\sigma}$, *and so* A *wins. To prove that* B *has a winning strategy in* $\mathcal{C}$ *we proceed similarly, by choosing the eager strategy* $\Sigma^!_\mathsf{B}$ *for* B.

We now show that $\mathcal{C}_\mathsf{A}$ *protects* A. *Let* $\mathcal{C}'$ *be composable with* $\mathcal{C}_\mathsf{A}$. *The eager strategy* $\Sigma^!_\mathsf{A}$ *is non-losing for* A. *Indeed, in every fair play* σ *conform to* $\Sigma^!_\mathsf{A}$, *if there exists* $\overline{e_i} \in R_\mathsf{A} \not\subseteq \overline{\sigma}$ *then* $e_{i+1} \in O_\mathsf{A} \not\in \overline{\sigma}$, *and so* $\Phi\mathsf{A}\sigma \geq 0$. *To prove that* $\mathcal{C}_\mathsf{B}$ *protects* B, *we proceed similarly, by choosing the eager strategy* $\Sigma^!_\mathsf{B}$ *for* B. □

7. Related work

Several papers address the problem of defining compliance over (various kinds of) behavioural types. This has given rise to many different notions of compliance: some of them, like the one in Definition 5.2, are shaped to guarantee the absence of deadlock [15, 1], while some others address different properties.

The notion of compliance proposed in [19] is based on *should-testing*: it requires that, from any reachable state, it is always possible to reach a success state. Note that this property is not guaranteed by progress-based notions. For instance, consider the session types:

$$P \;=\; \mathit{rec}\, X.\, (\mathtt{a?}.X + \mathtt{b?}.\mathbf{1}) \qquad\qquad Q \;=\; \mathit{rec}\, Y.\, \mathtt{a!}.Y$$

Here, Definition 5.2 states that P is compliant with Q, because $P \mid Q$ never deadlocks; however, P is *not* compliant with Q according to [19], because P can never reach the success state.

The notion of *I/O compliance* introduced in [20] addresses another issue, i.e. avoiding "vacuous" progress where P exposes some capabilities, Q cannot interact, and the composition $P \mid Q$ merely advances via internal transitions (without synchronisations). For instance, let:

$$P \;=\; (\mathit{rec}\, X.\, \mathtt{a!}.X)[] \qquad\qquad Q \;=\; \mathtt{b?}[]$$

where we use [] to denote an *asynchronous* semantics (via unbounded buffers). The process $P \mid Q$ never deadlocks, because P continues forever adding messages to the buffer. Thus, P would be compliant with Q according to progress-based compliance, while P is *not* I/O-compliant with Q. Whereas the overall framework of [20] abstracts from the process syntax, being defined over generic LTSs, in the special case of LTSs obtained through session types, *synchronous* I/O compliance is equivalent to progress-based compliance.

Our general encoding of LTSs into event structures (Definition 5.12) suggests that also other notions of compliance for session types (e.g. the above-mentioned ones) can be related to agreement in game-based contracts. Doing this would require to adjust the payoff function (the one in Definition 5.22 is specific for progress-based compliance), and by suitably restricting the class of admissible winning strategies. This approach to relate compliance to agreement is limited to those cases where compliance is formalised as a semantic property; it does not apply e.g. when compliance is defined as syntactic duality [21].

When compliance relates two behavioural types, there is a distinction between the *asymmetric* version, where only one of the two parties (the "client") is required to reach success, as in Definition 5.2, and the *symmetric* one, where both parties must succeed. A relation between these two variants is established in [22], which shows that asymmetric progress-based compliance can be defined in terms of the symmetric one.

In the *weak compliance* relation defined in [6], finite-state orchestrators can resolve external choices or rearrange messages in order to guarantee progress. Differently to strong (progress-based) compliance, weak compliance does not seem to be related to agreement. Consider e.g. the session types $P = \mathtt{a!} \oplus \mathtt{b!}$ and $Q = \mathtt{b?}$. While, by encoding these session types to contracts, we obtain an agreement through the strategy which tells A to fire $\mathtt{b!}$, P is *not* weakly compliant with Q because no orchestrator can prevent A from choosing the branch $\mathtt{a!}$. However, $P' = \mathtt{a!} + \mathtt{b!}$ (which is a legitimate term in [6]) is weakly compliant with Q, because the orchestrator can resolve the external non-determinism by choosing the branch $\mathtt{b!}$. One way to formalise weak compliance in game-based contracts would be modelling the orchestrator as a third player of the game (who can use *any* strategy to favour the interaction between A and B). This would also require adapting the construction of the contracts to take into account for the moves of the orchestrator.

Other notions of compliance address calculi featuring e.g. asynchronous communication via unbounded buffers [23, 20], and multi-party interactions [24, 25, 26]. Since our contracts are inherently multi-party, they induce a natural notion of compliance for multi-party session types: given $P_1, \ldots, P_n$, encode the LTS of $P_1 \mid \cdots \mid P_n$ into an ES via Definition 5.12, and then say that $P_1, \ldots, P_n$ are compliant whenever their eager strategies are winning, along the lines of Theorem 5.23. A relevant question would be that of finding an equivalent definition of multi-party compliance without passing through the encoding into ES.

Since agreement and protection are mutually exclusive in contracts based on ES (Theorem 6.11), in [27] a conservative extension of ES featuring *circular causality* (CES) is proposed to reconcile the two notions. There, circular dependencies are modelled with a new enabling relation $\Vdash$. The contract $a \Vdash b$ (intuitively, "I will do a if you *promise* to do b") reaches an agreement with the dual contract $b \Vdash a$, while protecting the participant who offers it. While in standard ES an event a which causally depends on b can only be performed *after* b, in CES the enabling $b \Vdash a$ allows a to happen before b, under the guarantee that b will be eventually performed. Using this refined model for contracts, a technique is proposed in [12] such that, starting from the participants O-R payoffs, constructs a set of contracts which protect their participants, and still admits an agreement.

In *contract-oriented computing* [17], interactions between services are driven by contracts. Participants advertise their contracts to some *contract brokers*, which are the contract-oriented analogous of service repositories in the Web Service paradigm. Participants wait until the contract broker finds an *agreement* among the contracts in its hands, and then they can start interacting (via a multi-party session), by performing the actions prescribed by their contracts. Differently from most of the approaches based on behavioural types [8, 28], a contract-oriented service is not supposed to be *honest*, in that it may not honour its contracts. The idea is that, if a service is not honest, then an external judge may inspect its contracts and the status of the session. In the case a violation is found, the judge will eventually provide the prescribed compensations/punishments. Some formal models for contract-oriented computing have been proposed, using as contracts logical formulae [14], process algebras [17], binary [29], and multi-party session types [30]. Since honesty is undecidable [29], some papers have addressed the problem of devising verification techniques to safely overapproximate it [31, 32]. In the setting of contract-oriented computing, the notion of protection studied in this paper addresses the issue of providing participants with non-losing strategies when the contract broker establishes sessions in the absence of an agreement.

The use of games to give semantics to programs and proofs is not new [33]. In particular, *concurrent* games were first used in [13] in order to define a model of MALL (multiplicative additive linear logic) enjoying *full completeness*. This property requires that not only formulae, but also proofs have an interpretation (a

morphism) in the model, and it guarantees that each morphism corresponds to an actual proof in the logic. Compared with sequential games, where a single player can move in a given state of the play, in concurrent games [13] *both* players can be in charge of doing a move in a given state. This also happens in our games, where, after a play σ, any player with an enabled event can perform it. However, our semantics of games is *interleaving* (i.e. a play is a plain sequence of events), while the one in [13] is true-concurrent. There is also a difference in the way strategies are formalised: in our work, strategies map game states (i.e. plays) to sets of events, while in [13] they map game states to game states. The strategies studied in [34] deal with games formalized as polarized event structures: for a game A, they are described in terms of a mapping between another event structure S and A. Such mapping, roughly, constrains the moves of A for both the Player and the Opponent. To avoid limiting the Opponent in an unreasonable way, further requirements are put on the strategy, namely *receptivity* and *innocence* (a distinct notion from ours). Our strategies are simpler, albeit less general: in our work, a strategy can be assigned to a specific participant, and mandates only the events performed by such participant; further, given a trace comprising the previously performed events, the strategy simply points out which moves the participant intends to do next. The rich mathematical structure of game states and of strategies is exploited e.g. in [13] to construct morphisms from strategies, as an intermediate step to prove full completeness. We do not need such a complex construction in our theory, as our objective is not to give semantics to programs or proofs, but just to use games to model interactions among mutually distrusted participants.

8. Conclusions

We have proposed a formal framework to represent and reason about contracts. In our setting, a contract is an event structure, used to model participants' obligations, and a payoff function, to model participants' objectives. A crucial notion is that of *agreement*, a property that guarantees safe interactions among participants. We have studied the meaning of agreement in the context of binary session types. In particular, we have shown that two session types are *compliant* if and only if their encodings to contracts have an agreement via an *eager* strategy (Theorem 5.23). We have then focussed on the property of *protection*. Differently from agreement, protection does not always guarantees safe interactions, but at least it allows a participant to obtain a non-negative payoff in arbitrary, possibly malicious, contexts. A main result is that, in a certain class of contracts, it is not possible to obtain at the same time agreement and protection for all participants (Theorem 6.11).

Our correspondence result about agreement in contracts and compliance in session types involves eager strategies, only. A still open 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 $\mathtt{a!} \oplus \mathtt{b!}$ 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 an example, 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 progress-based notion of compliance (Definition 5.2), the two session types:

$$P_{\mathsf{A}} \;=\; \mathtt{payCash!} \oplus \mathtt{payCC!} \qquad\qquad P_{\mathsf{B}} \;=\; \mathtt{payCash?}$$

are *not* compliant, and so (by Theorem 5.23) the eager strategy is *not* winning in $\mathcal{C}(P_{\mathsf{A}} \parallel P_{\mathsf{B}})$. A different interpretation of the internal choice of A would be the following: A is willing to choose between $\mathtt{payCash!}$ and $\mathtt{payCC!}$ 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 $\mathcal{C}(P_{\mathsf{A}} \parallel P_{\mathsf{B}})$ admits an agreement, via a non-eager strategy which requires A to renounce to the $\mathtt{payCC!}$ alternative.

Another question is whether the notion of protection is applicable to session types. The idea is that in some byzantine scenarios one participant may be forced to interact with the others in the absence of compliance. In this scenarios, the participant cannot aim at reaching success: instead, protection should provide her with a way to limit the damages. For instance, consider the session type:

$$P_{\mathsf{A}} = \mathtt{pay!.receive?} \oplus \mathtt{abort!}$$

If we make P_{A} interact with the session type $Q = \texttt{pay?}$ (with which P_{A} is not compliant), then A should avoid firing `pay!`, because doing so will never lead to the expected `receive?`. In this example, a strategy which protects A would be the one which only enables `abort!`. Defining protection for session types can be done, e.g. by assigning a payoff to each trace of a session type (composed with any context). In the above example, the payoff is zero for P_{A} and after `abort!`, it is positive after firing `receive?`, while it is negative after `pay!` but before `receive?`. By encoding of session types into contracts (Definition 5.22), all the results about contracts and protection can be exported also for session types. In particular, Theorem 6.11 will have as a consequence that compliance and protection are mutually exclusive when participants have finite circular O-R payoffs.

Our notion of contract slightly departs from the commonly accepted meaning of the word "contract", namely some entity which has been concretised after a process of "agreement", and which has after then become "legally binding". While we adhere to the principle that contracts are "legally binding", we also call contracts the entities used to reach an agreement. For instance, in our view a contract may be the statement made by a service through its Service Level Agreement — which indeed is a concrete entity even before any agreement is established.

To further motivate this choice, consider the terminology used in the domain of process algebras. There, both the atomic entities and their compositions are modelled as *processes*. For instance, both

$$\begin{aligned} P &= \bar{a}\langle v\rangle.\, b(x) && \text{(an output of } v \text{ on channel } a \text{ followed by an input on } b\text{)} \\ Q &= a(y).\, \bar{b}\langle y+1\rangle && \text{(an input on } a \text{ followed by an output on } b\text{)} \end{aligned}$$

are processes, as well as their composition $P \mid Q$. Now, assume that somehow there is an agreement between contracts P and Q. Then, $P \mid Q$ can be interpreted as a contract according to the common meaning. If we were going to accept the principle that contracts *exist* only after they have been agreed upon, then a process $P \mid Q'$, where e.g. $Q' = a(x).\, \bar{c}\langle x+1\rangle$ (the output is on the "wrong" channel) would not even exist as a contract. In our theory, we can reason about contracts before, or even in the absence of, an agreement. This allows us to understand what happens when a service advertises its contract in an environment populated by malicious adversaries.

References

[1] G. Castagna, N. Gesbert, L. Padovani, A theory of contracts for web services, ACM TOPLAS 31 (5) (2009) 19:1–19:61. doi:10.1145/1538917.1538920.

[2] F. Barbanera, U. de'Liguoro, Two notions of sub-behaviour for session-based client/server systems, in: Proc. PPDP, 2010, pp. 155–164. doi:10.1145/1836089.1836109.

[3] M. Bravetti, G. Zavattaro, Contract based multi-party service composition, in: Proc. FSEN, Vol. 4767 of LNCS, 2007, pp. 207–222. doi:10.1007/978-3-540-75698-9_14.

[4] W. M. P. van der Aalst, N. Lohmann, P. Massuthe, C. Stahl, K. Wolf, Multiparty contracts: Agreeing and implementing interorganizational processes, Comput. J. 53 (1) (2010) 90–106. doi:10.1093/comjnl/bxn064.

[5] F. Barbanera, U. de'Liguoro, Loosening the notions of compliance and sub-behaviour in client/server systems, in: Proc. ICE, Vol. 166 of EPTCS, Open Publishing Association, 2014, pp. 94–110. doi:10.4204/EPTCS.166.10.

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

[7] G. Winskel, Event structures, in: Advances in Petri Nets, 1986, pp. 325–392. doi:10.1007/3-540-17906-2_31.

[8] K. Honda, Types for dyadic interaction, in: Proc. CONCUR, 1993, pp. 509–523. doi:10.1007/3-540-57208-2_35.

[9] K. Honda, V. T. Vasconcelos, M. Kubo, Language primitives and type disciplines for structured communication-based programming, in: Proc. ESOP, 1998, pp. 122–138. doi:10.1007/BFb0053567.

[10] M. Nielsen, G. D. Plotkin, G. Winskel, Petri nets, event structures and domains, part I, Theoretical Computer Science 13 (1981) 85–108. doi:10.1016/0304-3975(81)90112-2.

[11] G. Winskel, An introduction to event structures, in: REX Workshop, 1988, pp. 364–397. doi:10.1007/BFb0013026.

[12] M. Bartoletti, T. Cimoli, R. Zunino, A theory of agreements and protection, in: Proc. POST, Vol. 7796 of LNCS, Springer, 2013, pp. 186–205. doi:10.1007/978-3-642-36830-1_10.

[13] S. Abramsky, P.-A. Mellies, Concurrent games and full completeness, in: Proc. LICS, 1999, pp. 431–431. doi:10.1109/LICS.1999.782638.

[14] M. Bartoletti, R. Zunino, A calculus of contracting processes, in: Proc. LICS, 2010, pp. 332–341. doi:10.1109/LICS.2010.25.

[15] C. Laneve, L. Padovani, The *must* Preorder Revisited, in: Proc. CONCUR, 2007, pp. 212–225. doi:10.1007/978-3-540-74407-8_15.

[16] D. König, Sur les correspondances multivoques des ensembles, Fundamenta Mathematicae 8 (1926) 114–134.
[17] M. Bartoletti, E. Tuosto, R. Zunino, Contract-oriented computing in CO_2, Scientific Annals in Computer Science 22 (1) (2012) 5–60. doi:10.7561/SACS.2012.1.5.
[18] S. Even, Y. Yacobi, Relations among public key signature systems, Tech. Rep. 175, Computer Science Department, Technion, Haifa (1980).
[19] M. Bravetti, G. Zavattaro, Towards a unifying theory for choreography conformance and contract compliance, in: Software Composition, 2007, pp. 34–50. doi:10.1007/978-3-540-77351-1_4.
[20] M. Bartoletti, A. Scalas, R. Zunino, A semantic deconstruction of session types, in: Proc. CONCUR, 2014, pp. 402–418. doi:10.1007/978-3-662-44584-6_28.
[21] G. Bernardi, O. Dardha, S. Gay, D. Kouzapas, On duality relations for session types, in: Proc. TGC, 2014, pp. 51–66. doi:10.1007/978-3-662-45917-1_4.
[22] M. Bugliesi, D. Macedonio, L. Pino, S. Rossi, Compliance preorders for web services, in: Proc. WS-FM, 2009, pp. 76–91. doi:10.1007/978-3-642-14458-5_5.
[23] M. Bravetti, G. Zavattaro, Contract compliance and choreography conformance in the presence of message queues, in: WS-FM, 2008, pp. 37–54. doi:10.1007/978-3-642-01364-5_3.
[24] P.-M. Deniélou, N. Yoshida, Multiparty compatibility in communicating automata: Characterisation and synthesis of global session types, in: Proc. ICALP, 2013, pp. 174–186. doi:10.1007/978-3-642-39212-2_18.
[25] J. Lange, E. Tuosto, N. Yoshida, From communicating machines to graphical choreographies, in: Proc. POPL, 2015, pp. 221–232. doi:10.1145/2676726.2676964.
[26] S. Basu, T. Bultan, M. Ouederni, Deciding choreography realizability, in: Proc. POPL, 2012, pp. 191–202. doi:10.1145/2103656.2103680.
[27] M. Bartoletti, T. Cimoli, G. M. Pinna, R. Zunino, Circular causality in event structures, Fundam. Inform. 134 (3-4) (2014) 219–259. doi:10.3233/FI-2014-1101.
[28] K. Honda, N. Yoshida, M. Carbone, Multiparty asynchronous session types, in: Proc. POPL, 2008, pp. 273–284. doi:10.1145/1328438.1328472.
[29] M. Bartoletti, E. Tuosto, R. Zunino, On the realizability of contracts in dishonest systems, in: Proc. COORDINATION, 2012, pp. 245–260. doi:10.1007/978-3-642-30829-1_17.
[30] M. Bartoletti, J. Lange, A. Scalas, R. Zunino, Choreographies in the wild, Science of Computer Programmingdoi:10.1016/j.scico.2014.11.015.
[31] M. Bartoletti, A. Scalas, E. Tuosto, R. Zunino, Honesty by typing, in: Proc. FORTE, 2013, pp. 305–320. doi:10.1007/978-3-642-38592-6_21.
[32] M. Bartoletti, M. Murgia, A. Scalas, R. Zunino, Modelling and verifying contract-oriented systems in Maude, in: Proc. WRLA, 2014, pp. 130–146. doi:10.1007/978-3-319-12904-4_7.
[33] S. Abramsky, G. McCusker, Game semantics, in: Computational Logic: Proceedings of the 1997 Marktoberdorf Summer School, Springer-Verlag, 1999, pp. 1–56.
[34] S. Rideau, G. Winskel, Concurrent strategies, in: Proc. LICS, 2011, pp. 409–418. doi:10.1109/LICS.2011.13.

A. Supplementary material and proofs

A.1. Proof of Lemma 5.10

Consider the relation $\mathcal{R}$ as defined at page 17:

$$\begin{aligned}
\mathcal{R} &= \{(P \mid Q,\ P \parallel Q) \mid P, Q \text{ session types}\} \cup \mathcal{R}_{\mathsf{A}} \cup \mathcal{R}_{\mathsf{B}} \\
\mathcal{R}_{\mathsf{A}} &= \{(\mathtt{a!}.P \mid Q,\ [\mathtt{a!}]P \parallel Q), \quad (\mathtt{a!}.P \mid \mathtt{b!}.Q',\ [\mathtt{a!}]P \parallel \mathtt{b!}.Q' \oplus Q'') \mid P, Q, Q', Q'' \text{ session types}\} \\
\mathcal{R}_{\mathsf{B}} &= \{(P \mid \mathtt{a!}.Q,\ P \parallel [\mathtt{a!}]Q), \quad (\mathtt{a!}.P' \mid \mathtt{b!}.Q,\ \mathtt{a!}.P' \oplus P'' \parallel [\mathtt{b!}]Q) \mid P, P', P'', Q \text{ session types}\}
\end{aligned}$$

We will prove that $\mathcal{R}$ is a turn-bisimulation for $s_1 = P \mid Q$ and $s_2 = P \parallel Q$ by showing, first, that $\mathcal{R}$ is a turn-simulation for s_1 and s_2 in *Part A*, and then, that $\mathcal{R}^{-1}$ is a turn-simulation for s_2 and s_1 in *Part B*.

Within each part, we proceed by cases on the form of s_1 and s_2; and for each case, we show that items (a), (b),(c) of Definition 5.8 hold. All the symmetric cases are omitted.

Part A:

Case 1: Let $s_1 = P \mid Q$ and $s_2 = P \parallel Q$, for P, Q session types.

(a)
1. $P \mid Q \to \mathtt{a!}.P' \mid Q$ if $P \to \mathtt{a!}.P'$, which implies $P = \mathtt{a!}.P' \oplus P''$. Hence, since $s_2 = P \parallel Q$ we have $\mathtt{a!}.P' \oplus P'' \parallel Q \xrightarrow{\mathsf{A}:\mathtt{a!}}\!\!\!\!\rightarrow\ [\mathtt{a!}]P' \parallel Q$. By definition of $\mathcal{R}$, we have that $(\mathtt{a!}.P' \mid Q,\ [\mathtt{a!}]P' \parallel Q) \in \mathcal{R}$.
2. $P \mid Q \to P' \mid Q'$ if $P \xrightarrow{\mathtt{a!}} P'$ and $Q \xrightarrow{\mathtt{a}} Q'$, which implies $P = \mathtt{a!}.P$ and $Q = \mathtt{a}.Q' + Q''$. Hence, since $s_2 = P \parallel Q$ we have $\mathtt{a!}.P \parallel Q \xrightarrow{\mathsf{A}:\mathtt{a!}}\!\!\!\!\rightarrow\ [\mathtt{a!}]P' \parallel \mathtt{a}.Q' + Q'' \xrightarrow{\mathsf{B}:\mathtt{a}}\!\!\!\!\rightarrow\ P' \parallel Q'$. Since P', Q' are session types, by definition of $\mathcal{R}$, we have that $(P' \mid Q',\ P' \parallel Q') \in \mathcal{R}$.
3. $P \mid Q \to P' \mid Q$, if $P \to \mathbf{0}$, which implies $P = \mathbf{1}$. Hence, since $s_2 = \mathbf{1} \parallel Q$, we have $\mathbf{1} \parallel Q \xrightarrow{\mathsf{A}:\checkmark}\!\!\!\!\rightarrow\ \mathbf{0} \parallel Q$. By definition of $\mathcal{R}$, we have that $(\mathbf{0} \mid Q,\ \mathbf{0} \parallel Q) \in \mathcal{R}$.

(b)
1. $P \parallel Q \xrightarrow{\mathsf{A}:\mathtt{a!}}\!\!\!\!\rightarrow\ [\mathtt{a!}]P' \parallel Q$ which implies either (i) $P = \mathtt{a!}.P' \oplus P''$ or (ii) $P = \mathtt{a!}.P'$ with $\nu \implies g$. In the first case (i), if $P = \mathtt{a!}.P' \oplus P''$ we have $P \mid Q \to \mathtt{a!}.P' \mid Q$. Hence, we proved that also $P \mid Q$ can move. In the second case (ii), if $P = \mathtt{a!}.P'$, for $P \mid Q$ to move, we must consider Q: if $Q = \mathtt{a}.Q' + Q''$ then $P \mid Q \to P' \mid Q'$. If $Q = \mathbf{1}$ then $P \mid Q \to P \mid \mathbf{0}$. Otherwise, $P \mid Q$ is stuck, but so is $\mathtt{a!}.P' \mid Q$ and by definition of $\mathcal{R}$, we have $(\mathtt{a!}.P' \mid Q,\ [\mathtt{a!}]P' \mid Q) \in \mathcal{R}$. Hence (b) is proved.
2. $P \parallel Q \xrightarrow{\mathsf{A}:\mathtt{a}}\!\!\!\!\rightarrow$. This case would require $P = [\mathtt{a!}]P'$ but it does not apply here since we are under the hypothesis of **case 1** and both P and Q are session types.
3. $P \parallel Q \xrightarrow{\mathsf{A}:\checkmark}\!\!\!\!\rightarrow\ \mathbf{0} \parallel Q$ if $P = \mathbf{1}$. Hence, $\mathbf{1} \mid Q \to \mathbf{0} \mid Q$.

(c) To prove (c), let us assume $s_2 \in S_2$, which implies $s_2 = \mathbf{0} \parallel \mathbf{0}$. Then by hypothesis of **case 1** we have $s_1 = \mathbf{0} \mid \mathbf{0} \in S_1$, which satisfies $s_1 \in S_1$.

Case 2: Let $s_1 = \mathtt{a!}.P' \mid Q$ and $s_2 = [\mathtt{a!}]P' \parallel Q$, for P', Q session types.

(a)
1. $\mathtt{a!}.P' \mid Q \to \mathtt{a!}.P' \mid \mathtt{b!}.Q'$ if $Q \to \mathtt{b!}.Q'$, which implies $Q = \mathtt{b!}.Q' + Q''$. Hence, $s_2 = [\mathtt{a!}]P' \parallel \mathtt{b!}.Q' + Q''$ is stuck, but already in relation $\mathcal{R}$ with $\mathtt{a!}.P' \mid \mathtt{b!}.Q'$.
2. $\mathtt{a!}.P' \mid Q \to P' \mid Q'$ if $Q = \mathtt{a}.Q' + Q''$. Hence, $[\mathtt{a!}]P' \parallel \mathtt{a}.Q' + Q'' \xrightarrow{\mathsf{B}:\mathtt{a}}\!\!\!\!\rightarrow\ P' \parallel Q'$. By definition of $\mathcal{R}$, we have that $(P' \mid Q', P' \parallel Q') \in \mathcal{R}$.
3. $\mathtt{a!}.P' \mid Q \to \mathtt{a!}.P' \mid Q'$ if $Q = \mathbf{1}$. Hence, $[\mathtt{a!}]P' \parallel Q \xrightarrow{\mathsf{B}:\checkmark}\!\!\!\!\rightarrow\ [\mathtt{a!}]P' \parallel \mathbf{0}$. By definition of $\mathcal{R}$, we have that $(\mathtt{a!}.P' \mid \mathbf{0},\ [\mathtt{a!}]P' \parallel \mathbf{0}) \in \mathcal{R}$.

(b)
1. $[\mathtt{a!}]P' \parallel Q \xrightarrow{\mathsf{B}:\mathtt{a!}}\!\!\!\!\rightarrow$. Not Possible.
2. $[\mathtt{a!}]P' \parallel Q \xrightarrow{\mathsf{B}:\mathtt{a}}\!\!\!\!\rightarrow\ P' \parallel Q'$ if $Q = \mathtt{a}.Q' + Q''$. Hence, $\mathtt{a!}.P' \mid \mathtt{a}.Q' + Q'' \to P' \mid Q'$.
3. $[\mathtt{a!}]P' \parallel Q \xrightarrow{\mathsf{B}:\checkmark}\!\!\!\!\rightarrow\ [\mathtt{a!}]P' \parallel \mathbf{0}$ if $Q = \mathbf{1}$. Hence $\mathtt{a!}.P' \mid \mathbf{1} \to \mathtt{a!}.P' \mid \checkmark$.

(c) To prove (c), let us assume $s_2 \in S_2$, which implies $s_2 = \mathbf{0} \mid \mathbf{0}$. By hypothesis of **case 2** this is not possible.

Case 3: Let $s_1 = \mathtt{a!}.P' \mid \mathtt{b!}.Q'$ and $s_2 = [\mathtt{a!}]P' \parallel \mathtt{b!}.Q' + Q''$, for P', Q', Q'' session types.

(a)
1. $\mathtt{a!}.P' \mid Q \to P' \mid Q'$ if $Q = \mathtt{a}.Q' + Q''$. Not possible.
2. $\mathtt{a!}.P' \mid Q \to \mathtt{a!}.P' \mid Q'$ if $Q = \mathbf{1}$. Not possible.
3. $\mathtt{a!}.P' \mid Q \to \mathtt{a!}.P' \mid \mathtt{b!}.Q'$ if $Q \to \mathtt{b!}.Q'$, which implies $Q = \mathtt{b!}.Q' + Q''$. Not possible.

(b)
1. $[\mathtt{a!}]P' \parallel \mathtt{b!}.Q' + Q'' \xrightarrow{\mathtt{B}:\mathtt{a!}}\!\!\!\!\rightarrow [\mathtt{a!}]P' \parallel [\mathtt{b!}].Q'$. Then we have s_1 stuck, but also $(s_1, [\mathtt{a!}]P' \parallel [\mathtt{b!}].Q') \in \mathcal{R}$ and $[\mathtt{a!}]P' \parallel [\mathtt{b!}].Q' \not\to$.
2. $[\mathtt{a!}]P' \parallel \mathtt{b!}.Q' + Q'' \xrightarrow{X:\mathtt{a}}\!\!\!\!\rightarrow$ Not possible for any $X \in \{A, B\}$.
3. $[\mathtt{a!}]P' \parallel \mathtt{b!}.Q' + Q'' \xrightarrow{X:\checkmark}\!\!\!\!\rightarrow$ Not possible for any $X \in \{A, B\}$.

(c) To prove (c), let us assume $s_2 \in S_2$, which implies $s_2 = \mathbf{0} \mid \mathbf{0}$. By hypothesis of **case 3**, this is not possible.

Part B:

Case 1: Let $s_1 = P \mid Q$ and $s_2 = P \parallel Q$, for P, Q session types.

(a)
1. $P \parallel Q \xrightarrow{\mathtt{A}:\mathtt{a!}}\!\!\!\!\rightarrow [\mathtt{a!}]P' \parallel Q$ which implies either (i) $P = \mathtt{a!}.P' \oplus P''$ or (ii) $P = \mathtt{a!}.P'$. In the first case (i), we have $P \mid Q \to \mathtt{a!}.P' \mid Q$ and by definition of $\mathcal{R}$ we have $(\mathtt{a!}.P' \mid Q,\ [\mathtt{a!}]P' \parallel Q) \in \mathcal{R}$. In the second case (ii), $\mathtt{a!}.P' \mid Q$ is already in relation $\mathcal{R}$ with $[\mathtt{a!}]P' \parallel Q$.
2. $P \parallel Q \xrightarrow{\mathtt{A}:\mathtt{a}}\!\!\!\!\rightarrow$. This case would require $Q = [\mathtt{a!}]Q'$ but it does not apply here since we are under the hypothesis of **case 1** and both P and Q are session types.
3. $P \parallel Q \xrightarrow{\mathtt{A}:\checkmark}\!\!\!\!\rightarrow \mathbf{0} \parallel Q$ if $Q = \mathbf{1}$. Hence, $P \mid \mathbf{1} \to P \mid \mathbf{0}$ and by definition of $\mathcal{R}$, $(P \mid \mathbf{0}, P \parallel \mathbf{0}) \in \mathcal{R}$.

(b)
1. $P \mid Q \to P' \mid Q$ if $P \to \mathtt{a!}.P'$, which implies $P = \mathtt{a!}.P' \oplus P''$. Hence, we have $\mathtt{a!}.P' \oplus P'' \parallel Q \xrightarrow{\mathtt{A}:\mathtt{a!}} \to [\mathtt{a!}]P' \parallel Q$. So, s_2 moves.
2. $P \mid Q \to P' \mid Q'$ if $P \xrightarrow{\mathtt{a!}} P'$ and $Q \xrightarrow{\mathtt{a}} Q'$, which implies $P = \mathtt{a!}.P'$ and $Q = \mathtt{a}.Q' + Q''$. Hence, we have $\mathtt{a!}.P' \parallel \mathtt{a}.Q' + Q'' \xrightarrow{\mathtt{A}:\mathtt{a!}}\!\!\!\!\rightarrow$. So, s_2 moves.
3. $P \mid Q \to P' \mid Q$, if $P \to \mathbf{0}$, which implies $P = \mathbf{1}$. Hence, we have $\mathbf{1} \parallel Q \xrightarrow{\mathtt{A}:\checkmark}\!\!\!\!\rightarrow \mathbf{0} \parallel Q$. So, s_2 moves.

(c) To prove (c), let us assume $s_1 \in S_1$, which implies $s_1 = \mathbf{0} \mid \mathbf{0}$. Then by hypothesis of **case 1** we have $s_2 = \mathbf{0} \parallel \mathbf{0} \in S_2$.

Case 2: Let $s_1 = \mathtt{a!}.P' \mid Q$ and $s_2 = [\mathtt{a!}]P' \parallel Q$, for P', Q session types.

(a)
1. $[\mathtt{a!}]P' \parallel Q \xrightarrow{\mathtt{B}:\mathtt{a!}}\!\!\!\!\rightarrow$. Not Possible.
2. $[\mathtt{a!}]P' \parallel Q \xrightarrow{\mathtt{B}:\mathtt{a}}\!\!\!\!\rightarrow\ P' \parallel Q'$ if $Q = \mathtt{a}.Q' + Q''$. Hence, we have $\mathtt{a!}.P \mid Q \to P' \mid Q'$ and by definition of $\mathcal{R}$, we have $(P' \mid Q', P' \parallel Q') \in \mathcal{R}$.
3. $[\mathtt{a!}]P' \parallel Q \xrightarrow{\mathtt{B}:\checkmark}\!\!\!\!\rightarrow\ [\mathtt{a!}]P' \parallel \mathbf{0}$ if $Q = \mathbf{1}$. Hence $\mathtt{a!}.P' \mid \mathbf{1} \to \mathtt{a!}.P' \mid \checkmark$ and by definition of $\mathcal{R}$, we have $(P' \mid \checkmark,\ P' \parallel \checkmark) \in \mathcal{R}$.

(b)
1. $\mathtt{a!}.P' \mid Q \to \mathtt{a!}.P' \mid \mathtt{b!}.Q'$ if $Q = \mathtt{b!}.Q' + Q''$. Hence, $[\mathtt{a!}]P' \parallel Q \not\to\!\!\!\!\rightarrow$ but $\mathtt{a!}.P' \mid \mathtt{b!}.Q'\ \mathcal{R}\ [\mathtt{a!}]P' \mid \mathtt{b!}.Q' + Q''$ and $\mathtt{a!}.P' \mid \mathtt{b!}.Q' \not\to$.
2. $\mathtt{a!}.P' \mid Q \to P' \mid Q'$ if $Q = \mathtt{a}.Q' + Q''$. Hence, $[\mathtt{a!}]P' \parallel Q \xrightarrow{\mathtt{B}:\mathtt{a}}\!\!\!\!\rightarrow\ P' \parallel Q'$.
3. $\mathtt{a!}.P' \mid Q \to \mathtt{a!}.P' \mid \mathbf{0}$ if $Q = \mathbf{1}$. Hence, $[\mathtt{a!}]P' \parallel Q \xrightarrow{\mathtt{B}:\checkmark}\!\!\!\!\rightarrow\ [\mathtt{a!}]P' \parallel \mathbf{0}$.

(c) To prove (c),let us assume $s_2 \in S_2$, which implies $s_2 = \mathbf{0} \mid \mathbf{0}$. By hypothesis of **case 2**, this is not possible.

Case 3: Let $s_1 = \mathtt{a!}.P' \mid \mathtt{b!}.Q'$ and $s_2 = [\mathtt{a!}]P' \parallel \mathtt{b!}.Q' + Q''$, for P', Q', Q'' session types.

(a) 1. $[\mathtt{a!}]P' \parallel \mathtt{b!}.Q' + Q'' \xrightarrow{\mathtt{B}:\mathtt{b!}}\!\!\!\!\rightarrow [\mathtt{a!}]P' \parallel [\mathtt{b!}]Q'$. Hence we have $(s_1, [\mathtt{a!}]P' \parallel [\mathtt{b!}]Q') \in \mathcal{R}$.

2. $[\mathtt{a!}]P' \parallel \mathtt{b!}.Q' + Q'' \xrightarrow{X:\mathtt{c}}\!\!\!\!\rightarrow$. Not possible for any X and c.

3. $[\mathtt{a!}]P' \parallel \mathtt{b!}.Q' + Q'' \xrightarrow{X:\checkmark}\!\!\!\!\rightarrow$. Not possible for any X.

(b) To prove (b) for **case 3**, we must check all the possible moves of $\mathtt{a!}.P' \mid \mathtt{b!}.Q' \rightarrow$, which is stuck.

(c) To prove (c), let us assume $s_1 \in S_1$, which implies $s_1 = \mathbf{0} \mid \mathbf{0}$. By hypothesis of **case 3**, this is not possible. □

A.2. Labelled Transition Systems over event structures

In this section we introduce an alternative LTS for event structures, which is based on the notion of *remainder* (Definition A.1). This will be needed later on in the proof of Theorem 5.20.

Given an event $e \in \mathbf{E}$ and an ES $\mathcal{E} = (E, \#, \vdash, \ell)$, we introduce the notion of *remainder* of $\mathcal{E}$ and e. This is another ES, denoted by $\mathcal{E}[e]$, where the event e is considered as occurred. The intuition is that the remainder $\mathcal{E}[e]$ is an event structure whose configurations C are such that $C \cup \{e\}$ is a configuration of $\mathcal{E}$.

Definition A.1 (Remainder of an ES). *For all* ES*s* $\mathcal{E} = (E, \#, \vdash, \ell)$ *and for all* $e \in \mathbf{E}$*, we define the* ES $\mathcal{E}[e]$ *as* $(E', \#', \vdash', \ell')$*, where:*

$$\begin{array}{rcl} E' & = & E \setminus (\{e\} \cup \{e' \mid e \# e'\}) \\ \#' & = & \# \cap (E' \times E') \\ \vdash' & = & \{(X \setminus \{e\}, e') \mid X \vdash e' \ \wedge \ X \subseteq E' \cup \{e\} \ \wedge \ e' \in E'\} \\ \ell' & = & \ell|_{E'} \end{array}$$

Further, for all $\sigma = \langle e_1 \cdots e_n \rangle$*, we define* $\mathcal{E}[\sigma]$ *as* $\mathcal{E}[e_1] \cdots [e_n]$.

The events of the remainder of $\mathcal{E}$ and e are those of $\mathcal{E}$ without e and all the events that are in conflict with e. According to the intuition, the enablings of $\mathcal{E}[e]$ are obtained by the enablings $X \vdash e'$ of $\mathcal{E}$ with $e \in X$: as the configuration C of $\mathcal{E}[e]$ must be such that $C \cup \{e\}$ is a configuration of $\mathcal{E}$, we have to be sure that only events that depend on e in $\mathcal{E}$ are enabled in $\mathcal{E}[e]$.

By Definition A.1, it immediately follows that $\mathcal{E}[e] = \mathcal{E}$ whenever $e \notin E$. This observation leads to the fact that we can calculate $\mathcal{E}[e]$ without requiring that e can be actually *fired* in $\mathcal{E}$, even when $e \in E$.

The notion of remainder naturally induces a Labelled Transition System (LTS) over event structures, representing their sequential computations. The states of this LTS are event structures, the labels are events, and the transition relation contains $\mathcal{E} \xrightarrow{e} \mathcal{E}'$ whenever $\mathcal{E}' = \mathcal{E}[e]$ for some event e immediately enabled in $\mathcal{E}$. We will show in Lemma A.5 below that this LTS is equivalent to the one over configurations (Definition 2.9).

Definition A.2 (LTS over ES). *We define the LTS* $(\mathbf{ES}, \mathbf{E}, \rightarrow)$ *with the following transition relation:*

$$\mathcal{E} \xrightarrow{e} \mathcal{E}[e] \qquad \text{if } \vdash e \ \in \mathcal{E}$$

The following lemma establishes a confluence result, namely: given a set of fired events, the order in which we pick them to build the remainder is irrelevant.

Lemma A.3. *Let* $\mathcal{E} = (E, \#, \vdash, \ell)$*, and let* $a, b \in E$ *be such that* $\neg(a \# b)$*. Then,* $\mathcal{E}[a][b] = \mathcal{E}[b][a]$.

Proof. The case $a = b$ is trivial. Let $a \neq b$. According to Definition A.1, we have $\mathcal{E}[a] = (E', \#', \vdash', \ell')$ with

$$\begin{array}{rcl} E' & = & E \setminus (\{a\} \cup \{e' \mid a \# e'\}) \\ \#' & = & \# \cap (E' \times E') = \{(e, e') \mid e \# e' \wedge e \in E' \wedge e' \in E'\} \\ \vdash' & = & \{(X \setminus \{a\}, e') \mid X \vdash e' \ \wedge \ X \subseteq (E' \cup \{a\}) \ \wedge \ e' \in E'\} \\ \ell' & = & \ell|_{E'} \end{array}$$

Then we have $\mathcal{E}[a][b] = (E'', \#'', \vdash'', \ell'')$, with

$$\begin{array}{rcl} E'' & = & E' \setminus (\{b\} \cup \{e' \mid b \#' e'\}) \\ \#'' & = & \#' \cap (E'' \times E'') = \{(e, e') \mid e \#' e' \wedge e \in E'' \wedge e' \in E''\} \\ \vdash'' & = & \{(X \setminus \{b\}, e') \mid X \vdash' e' \ \wedge \ X \subseteq (E'' \cup \{b\}) \ \wedge \ e' \in E''\} \\ \ell'' & = & \ell|_{E''} \end{array}$$

Since $\#$ is irreflexive and symmetric, and since $\neg(a \# b)$ and $a \neq b$ we have:

$$\begin{array}{rcl} E'' & = & E' \setminus (\{b\} \cup \{e' \mid b \#' e'\}) \\ & = & E \setminus (\{a\} \cup \{e' \mid a \# e'\} \cup \{b\} \cup \{e' \mid b \#' e'\}) \\ & = & E \setminus (\{a, b\} \cup \{e' \mid a \# e'\} \cup \{e' \mid b \# e' \wedge b \in E' \wedge e' \in E'\}) \\ & = & E \setminus (\{a, b\} \cup \{e' \mid a \# e'\} \cup \{e' \mid b \# e' \wedge (b \in E \wedge b \neq a \wedge \neg(a \# b)) \wedge (e' \in E \wedge e' \neq a \wedge \neg(a \# e'))\}) \\ & = & E \setminus (\{a, b\} \cup \{e' \mid a \# e'\} \cup \{e' \mid b \# e' \wedge (e' \neq a \wedge \neg(a \# e'))\}) \\ & = & E \setminus (\{a, b\} \cup \{e' \mid a \# e' \vee b \# e'\}) \end{array}$$

Hence, unfolding the definition of $\#''$, we have:

$$\begin{array}{rcl} \#'' & = & \#' \cap (E'' \times E'') = \{(e, e') \mid e \#' e' \wedge e \in E'' \wedge e' \in E''\} \\ & = & \{(e, e') \mid e \# e' \wedge e \in E' \wedge e' \in E' \wedge e \in E'' \wedge e' \in E''\} \\ & = & \{(e, e') \mid e \# e' \wedge \neg(e \# a) \wedge \neg(e \# b) \wedge \neg(e' \# a) \wedge \neg(e' \# b)\} \\ & = & \# \setminus \{(e, e') \mid (e \# a) \vee (e \# b) \vee (e' \# a) \vee (e' \# b)\} \end{array}$$

We can now unfold the definition of $\vdash''$, obtaining:

$$\begin{array}{rcl} \vdash'' & = & \{(X \setminus \{b\}, e') \mid (X, e') \in \vdash' \ \wedge \ X \subseteq (E'' \cup \{b\}) \ \wedge \ e' \in E''\} \\ & = & \{(X \setminus \{a, b\}, e') \mid (X, e') \in \vdash \ \wedge \ X \subseteq (E'' \cup \{a, b\}) \ \wedge \ e' \in E''\} \end{array}$$

Resuming, we have obtained: $\mathcal{E}[a][b] = (E'', \#'', \vdash'', \ell'')$ with

$$\begin{array}{rcl} E'' & = & E \setminus (\{a, b\} \cup \{e' \mid a \# e' \vee b \# e'\}) \\ \#'' & = & \# \setminus \{(e, e') \mid (e \# a) \vee (e \# b) \vee (e' \# a) \vee (e' \# b)\} \\ \vdash'' & = & \{(X \setminus \{a, b\}, e') \mid (X, e') \in \vdash \ \wedge \ X \subseteq (E'' \cup \{a, b\}) \ \wedge \ e' \in E''\} \\ \ell'' & = & \ell|_{E''} \end{array}$$

Since what we have obtained does not depend from the order of a and b, we have the thesis. □

By using Lemma A.3 in a simple inductive argument we obtain the following corollary: when computing the remainder of $\mathcal{E}[e_1 \cdots e_n]$, we can ignore the order of the events, provided that they are conflict-free. This allows us to use the shorthand $\mathcal{E}[C]$ for $\mathcal{E}[e_1 \cdots e_n]$ whenever $\{e_1, \ldots, e_n\} = C$ and $CF(C)$.

Corollary A.4. *Let $\mathcal{E} = (E, \#, \vdash, \ell)$, and let $\sigma \in E^*$ be such that $CF(\overline{\sigma})$. Then:*

(a) for all η such that $\overline{\sigma} = \overline{\eta}$, $\mathcal{E}[\sigma] = \mathcal{E}[\eta]$;

(b) $\mathcal{E}[\sigma] = (\hat{E}, \hat{\#}, \hat{\vdash}, \hat{\ell})$, where:

$$\begin{array}{rcl} \hat{E} & = & E \setminus (\overline{\sigma} \cup \{e' \mid \exists e \in \overline{\sigma}.\ e \# e'\}) \\ \hat{\#} & = & \# \setminus \{(e, e') \mid \exists e'' \in \overline{\sigma}.\ e \# e'' \ \vee \ e' \# e''\} \\ \hat{\vdash} & = & \{(X \setminus \overline{\sigma}, e') \mid X \vdash e' \ \wedge \ X \subseteq \hat{E} \cup \overline{\sigma} \ \wedge \ e' \in \hat{E}\} \\ \hat{\ell} & = & \ell|_{\hat{E}} \end{array}$$

The following lemma establishes a connection between the LTSs in Definitions A.2 and 2.9. Given an ES $\mathcal{E}$, the initial state $\emptyset$ of the configuration-based LTS $\rightarrow_{\mathcal{E}}$ is bisimilar to the state $\mathcal{E}$ of the ES-based LTS.

Lemma A.5. $(\emptyset, \rightarrow_{\mathcal{E}}) \sim (\mathcal{E}, \rightarrow)$.

Proof. Let $\mathcal{E} = (E, \#, \vdash, \ell)$, and let:

$$\mathcal{R} \;=\; \{(C, \mathcal{E}[C]) \mid C \subseteq_{fin} E \;\wedge\; C \in \mathcal{F}_{\mathcal{E}}\} \tag{14}$$

We will prove that $\mathcal{R}$ is a bisimulation. Let $(C, \mathcal{E}[C]) \in \mathcal{R}$. By item (b) of Corollary A.4, we have $\mathcal{E}[C] = (\hat{E}, \hat{\#}, \hat{\vdash}, \hat{\ell})$, with:

$$\hat{E} \;=\; E \setminus (C \cup \{e' \mid \exists a \in C.\; a \# e'\}) \tag{15}$$

$$\hat{\vdash} \;=\; \{(X \setminus C, e') \mid X \vdash e' \;\wedge\; X \subseteq \hat{E} \cup C \;\wedge\; e' \in \hat{E}\} \tag{16}$$

We have the following two cases:

- move of C. Assume that $C \xrightarrow{e} C \cup \{e\}$. By Definition 2.9 we have $C \vdash e$, $e \notin C$ and $CF(C \cup \{e\})$. Since $e \notin C$ and $CF(C \cup \{e\})$, by (15) we obtain $e \in \hat{E}$; and since $C \vdash e$, by (16) we obtain $(C \setminus C, e) = (\emptyset, e) \in \hat{\vdash}$. So, by $\hat{\vdash} e$ and by item (a) of Corollary A.4 it follows that $\mathcal{E}[C] \xrightarrow{e} \mathcal{E}[C][e] = \mathcal{E}[C \cup \{e\}]$. By definition of $\mathcal{R}$ in (14), we conclude that $(C \cup \{e\}, \mathcal{E}[C \cup \{e\}])$.

- move of $\mathcal{E}[C]$. Assume that $\mathcal{E}[C] \xrightarrow{e} \mathcal{E}[C][e]$. By Definition A.2 we have that $\hat{\vdash} e$. Since $e \in \hat{E}$, then by (15) we have $CF(C \cup \{e\})$ and $e \notin C$. Since $\hat{\vdash} e$, by (16) it must be $(X, e') \in \vdash$ for some X such that $X \subseteq \hat{E} \cup C$ and $X \setminus C = \emptyset$. This implies $X \subseteq C$, and by saturation we obtain $C \vdash e$. Since $C \vdash e$, $CF(C \cup \{e\})$ and $e \notin C$, by Definition 2.9 we obtain $C \xrightarrow{e}_{\mathcal{E}} C \cup \{e\}$. By (14), we conclude that $(C \cup \{e\}, \mathcal{E}[C \cup \{e\}])$. □

A.3. Constructions on event structures

We now review the operations on event structures, in order to prove Theorem 5.20 in Theorem 5.20.

The first construction is the lifting of an event structure. We add a new event to an ES and this one is the initial event. The resulting event structure is such that the added event is the only one enabled at the empty configuration and all the other events *depend* on this added one.

Definition A.6 (Lifting of an ES). *Let $\mathcal{E} = (E, \#, \vdash, \ell)$ be an* ES*, and let e be an event not in E. We define $(e, \alpha) \,.\, \mathcal{E}$ as the* ES *$(E, \#', sat(\vdash'), \ell')$, where:*

$$\begin{aligned} E' &= E \cup \{e\} \\ \#' &= \# \\ \vdash' &= \{(X \cup \{e\}, e') \mid (X, e') \in \vdash\} \cup \{(\emptyset, e)\} \\ \ell' &= \ell \cup \{(e, \alpha)\} \end{aligned}$$

The definition of sum of ESs, which models the choice, is standard. This operation is like the union of two event structure, except that the conflict relation is defined in such that, once that a choice has been made, all the alternatives are discarded.

Definition A.7 (Sum of two ESs). *Let $\mathcal{E}_1 = (E_1, \#_1, \vdash_1, \ell_1)$ and $\mathcal{E}_2 = (E_2, \#_2, \vdash_2, \ell_2)$ be two* ES*s such that $E_1 \cap E_2 = \emptyset$. We define their* sum *$\mathcal{E}_1 \boxplus \mathcal{E}_2$ as the* ES *$(E, \#, \vdash, \ell)$ where:*

$$\begin{aligned} E &= E_1 \cup E_2 \\ \# &= \#_1 \;\cup\; \#_2 \;\cup\; \{(e, e') \mid e \in E_i \;\wedge\; e' \in E \setminus E_i,\; \textit{with } i \in \{1, 2\}\} \\ \vdash &= \vdash_1 \;\cup\; \vdash_2 \\ \ell &= \ell_1 \;\cup\; \ell_2 \end{aligned}$$

Notice that here we do not have to require that the enablings of the resulting ES have to be saturated, as they are already saturated. The sum operation introduced above is clearly associative and commutative.

Lemma A.8. *The operation $\boxplus$ on ES is commutative and associative.*

In order to define the denotation turn-based configurations we need some auxiliary notions. The first one establishes when two enabling sets in two ESs can be *matched* whereas the second introduce an auxiliary conflict relations among two ESs, intuitively stating when two internal choices belonging to two different ESs are mutually exclusive. The pairs of events in this conflict relation are those that can be reached in the same number of steps in both ESs, provided that they are both labeled in $\mathcal{P}_\mathcal{U} \times (\mathtt{A}^! \cup \{\checkmark\})$.

Definition A.9. *Let E be a set of events, and let ℓ be a labeling function. We denote with $\dagger$ the minimal relation among subsets of events such that:*

- $\emptyset \dagger \emptyset$, *and*
- $X \cup \{e\} \dagger Y \cup \{e'\}$ *if* $\ell(e) = co(\ell(e'))$ *and* $X \dagger Y$.

Definition A.10 (Turn conflict of ESs). *Let $\mathcal{E}_1 = (E_1, \#_1, \vdash_1, \ell_1)$ and $\mathcal{E}_2 = (E_2, \#_2, \vdash_2, \ell_2)$ be two* ES*s such that $E_1 \cap E_2 = \emptyset$. We define the* turn-conflict *relation $\sharp$ as $\bigcup_{k \geq 0} \sharp^{(k)}$ where each $\sharp^{(k)}$ is as follows:*

$$\begin{aligned}
\sharp^{(0)} &= \{(e,e') \mid \emptyset \vdash_i e \ \wedge\ \emptyset \vdash_j e' \ \wedge\ \{\ell_i(e), \ell_j(e')\} \subseteq \mathcal{P}_\mathcal{U} \times \mathtt{A}^!\} \\
\sharp^{(k)} &= \{(e,e') \mid X \vdash_i e \ \wedge\ Y \vdash_j e' \ \wedge\ X \dagger Y \ \wedge\ \{\ell_i(e), \ell_j(e')\} \subseteq \mathcal{P}_\mathcal{U} \times \mathtt{A}^! \ \wedge\ X \cup Y \in Con^{(k-1)}\} \\
&\quad \cup\ \sharp^{(k-1)}
\end{aligned}$$

where $Con^{(k)} = \{X \subseteq_{fin} E_1 \cup E_2 \mid |X| = k \ \wedge\ \forall e, e' \in X : (e,e') \notin \sharp^{(k)}\}$, $i, j \in \{1,2\}$ and $i \neq j$.

Example A.11. *Consider the following two ESs $\mathcal{E}_1 = (E_1, \#_1, \vdash_1, \ell_1)$ and $\mathcal{E}_2 = (E_2, \#_2, \vdash_2, \ell_2)$ where $E_1 = \{e_1, e_3, e_5, e_7, e_9\}$, $E_2 = \{e_2, e_4, e_6, e_8, e_10, e_12\}$, the conflict relations are*

$$\#_1 = \left\{ \begin{array}{c} e_1\#_1e_5, e_1\#_1e_7, e_1\#_1e_9, \\ e_3\#_1e_5, e_3\#_1e_7, e_3\#_1e_9 \end{array} \right\} \qquad \#_2 = \left\{ \begin{array}{c} e_2\#_2e_8, e_2\#_2e_{10}, e_2\#_2e_{12}, e_4\#_2e_8, e_4\#_2e_{10} \\ e_4\#_2e_{12}, e_6\#_2e_8, e_6\#_2e_{10}, e_6\#_2e_{12} \end{array} \right\}$$

and the enablings of these ESs are

$$\vdash_1 = \left\{ \begin{array}{c} \vdash_1 e_1, e_1 \vdash_1 e_3, \\ \vdash_1 e_5, e_5 \vdash_1 e_7, \{e_5, e_7\} \vdash_1 e_9 \end{array} \right\} \qquad \vdash_2 = \left\{ \begin{array}{c} \vdash_2 e_2, e_2 \vdash_2 e_4, \{e_2, e_4\} \vdash_2 e_6 \\ \vdash_2 e_8, e_8 \vdash_2 e_{10}, \{e_8, e_{10}\} \vdash_2 e_{12} \end{array} \right\}$$

Furthermore assume that $\ell_1(e_1) = \ell_1(e_7) = \mathtt{A} : \mathtt{a!}$, $\ell_1(e_5) = \mathtt{A} : \mathtt{b!}$, and the others events in E_1 are labelled with $\mathtt{A} : \checkmark$, whereas $\ell_1(e_2) = \ell_2(e_{10}) = \mathtt{B} : \mathtt{a?}$, $\ell_\mathtt{B}(e_4) =: \mathtt{b!}$, $\ell_\mathtt{B}(e_8) = \mathtt{B} : \mathtt{b?}$, and the other events in E_2 are labelled $\mathtt{B} : \checkmark$.

The $\sharp$ relation contains the pair (e_7, e_4) as the first event structure contains the enabling $e_5 \vdash_1 e_7$, the second one the enabling $e_2 \vdash_2 e_4$, $\{e_5\} \dagger \{e_2\}$ and $\ell_1(e_7), \ell_2(e_4) \in \mathcal{P}_\mathcal{U} \times \mathtt{A}^!$.

We are now ready to introduce the main operation on event structures denoting session types. The event structure denoting the interaction among session types should mimic the capability of a session type to perform an internal choice, and of the other to react to this choice, if allowed. This is achieved by defining the enabling $\vdash'$. The difference in the treatment of internal and external choice is driven by the fact that in session types internal choice can always be performed whereas external ones are reactions to internal ones.

Definition A.12 (Turn composition of ES). *Let $\mathcal{E}_1 = (E_1, \#_1, \vdash_1, \ell_1)$ and $\mathcal{E}_2 = (E_2, \#_2, \vdash_2, \ell_2)$ be two* ES*s such that $E_1 \cap E_2 = \emptyset$, and let $E' \subseteq \{e \in E_1 \cup E_2 \} \vdash_i e \ \wedge\ \ell_i(e) \in \mathcal{P}_\mathcal{U} \times (\mathtt{A}^! \cup \{\checkmark\})$, with $i \in \{1,2\}$. Let*

$\sharp$ *be the relation on* $(E_1 \times E_2) \cup (E_2 \times E_1)$ *of Definition A.10 computed on* $\mathcal{E}_1$ *and* $\mathcal{E}_2$*. We define the* turn *composition* $\mathcal{E}_1 \boxdot \mathcal{E}_2$ *as the* ES $(E, \#, \vdash, \ell)$*, where:*

$$\begin{array}{rcl} E & = & (E_1 \cup E_2) \\ \# & = & (\#_1 \cup \#_2 \cup \sharp) \cap (E \times E) \\ \ell & = & (\ell_1 \cup \ell_2)|_E \end{array}$$

and $\vdash$ *is obtained by saturating the following relation:*

$$\begin{array}{rl} \vdash' = & \{(\emptyset, e) \mid e \in E' \wedge \vdash_i e)\} \cup \{(\{e\}, e') \mid e \in E' \wedge e' \in E \setminus E' \wedge \emptyset \vdash_i e' \wedge \ell_i(e') = co(\ell(e))\} \quad (a) \\ & \cup \{(X \cup Y, e) \mid X \neq \emptyset \wedge X \vdash_i e \wedge \ell(e) \in \mathcal{P}_{\mathcal{U}} \times (\mathtt{A}^! \cup \{\checkmark\}) \wedge \exists e''.\ Y \vdash_j e'' \wedge X \dagger Y \\ & \qquad \wedge X \cup Y \in Con\} \quad (b) \\ & \cup \{(X \cup Y \cup \{e'\}, e) \mid X \neq \emptyset \wedge X \vdash_i e \wedge \ell(e) \in \mathcal{P}_{\mathcal{U}} \times \mathtt{A}^? \wedge \exists e''.\ Y \vdash_j e'' \wedge X \dagger Y \\ & \qquad \wedge X \cup Y \cup \{e'\} \in Con \wedge e' \in E_j \wedge \ell(e') = co(\ell(e)) \wedge e' \notin X \cup Y\} \quad (c) \end{array}$$

where $i, j \in \{1, 2\}$ *and* $i \neq j$*.*

To obtain a new enabling in the compound event structure we have just one relevant condition, namely that there must exists two suitable enablings in both components ($X \vdash_i e$ and $Y \vdash_j e'$) and X can be matched with Y. This allows to establish that an enabling involving $X \cup Y$ and either e or e' should be in the enabling relation of the compound event structure. More precisely $X \cup Y \vdash e$ (or $X \cup Y \vdash e'$) is introduced when $\ell(e) \in \mathcal{P}_{\mathcal{U}} \times (\mathtt{A}^! \cup \{\checkmark\})$ ($\ell(e') \in \mathcal{P}_{\mathcal{U}} \times (\mathtt{A}^! \cup \{\checkmark\})$ respectively), which is the clause (b), whereas when $\ell(e) \in \mathcal{P}_{\mathcal{U}} \times \mathtt{A}^?$ or $\ell(e') \in \mathcal{P}_{\mathcal{U}} \times \mathtt{A}^?$ then e (e' respectively) has to be matched by a corresponding $e'' \notin X \cup Y$ such that $\ell(e'') = co(\ell(e))$ ($\ell(e'') = co(\ell(e))$ respectively), and this is the clause (c) of this definition.

Example A.13. *Consider again the two event structures of Example 5.21* $\mathcal{E}_\mathtt{A} = (E_\mathtt{A}, \#_\mathtt{A}, \vdash_\mathtt{A}, \ell_\mathtt{A})$ *and* $\mathcal{E}_\mathtt{B} = (E_\mathtt{B}, \#_\mathtt{B}, \vdash_\mathtt{B}, \ell_\mathtt{B})$*. The enablings of these events structures are*

$$\vdash_\mathtt{A} = \left\{ \begin{array}{c} \vdash_\mathtt{A} e_1, e_1 \vdash_\mathtt{A} e_3, \\ \vdash_\mathtt{A} e_5, e_5 \vdash_\mathtt{A} e_7, \{e_5, e_7\} \vdash_\mathtt{A} e_9 \end{array} \right\} \qquad \vdash_\mathtt{B} = \left\{ \begin{array}{c} \vdash_\mathtt{B} e_2, e_2 \vdash_\mathtt{B} e_4, \{e_2, e_4\} \vdash_\mathtt{B} e_6 \\ \vdash_\mathtt{B} e_8, e_8 \vdash_\mathtt{B} e_{10}, \{e_8, e_{10}\} \vdash_\mathtt{B} e_{12} \end{array} \right\}$$

The set E' *is* $\{e_1, e_5\}$*, and the enablings of* $\mathcal{E}_\mathtt{A} \boxdot \mathcal{E}_\mathtt{B}$ *are*

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

where, for instance, $\vdash e_1$ *is present as* $e_1 \in E'$*, and* $e_1 \vdash e_2$ *is present as* $e_1 \in E'$ *and* $\vdash_\mathtt{B} e_2$ *(in both cases the clause (a) of Definition A.12 is used),* $\{e_1, e_2\} \vdash e_3$ *is derived because* $e_1 \vdash_\mathtt{A} e_3$ *and* $e_2 \vdash_\mathtt{B} e_4$*, as* $\ell_\mathtt{A}(e_1) = co(\ell_\mathtt{B}(e_2))$*, and then* $\{e_1\} \dagger \{e_2\}$ *(clause (b) of Definition A.12). Finally* $\{e_5, e_7, e_8\} \vdash e_{10}$ *derives from* $e_8 \vdash_\mathtt{B} e_{10}$*,* $e_5 \vdash_\mathtt{A} e_7$*,* $\ell_\mathtt{A}(e_7) = co(\ell_\mathtt{B}(e_10))$ *and* $\{e_5\} \dagger \{e_8\}$ *(clause (c) of Definition A.12).*

The semantics of turn-based configurations involve an intermediate step that basically uses a one-position buffer which store the name of the action to be done. We have to specialize the definition of turn composition to this peculiar situation.

Definition A.14 (Buffered turn composition of ES). *Let* $\mathcal{E}_1 = \langle E_1, \#_1, \vdash_1, \ell_1 \rangle$ *and* $\mathcal{E}_2 = \langle E_2, \#_2, \vdash_2, \ell_2 \rangle$ *be two ES such that* $E_1 \cap E_2 = \emptyset$*, let* $\mathtt{a?} \in \mathtt{A}$ *and* $\hbar \in \{1, 2\}$*. Let* $E' = \{e \in E_\hbar \mid \vdash_\hbar e \wedge \ell_\hbar(e) \in \mathcal{P}_{\mathcal{U}} \times \{\mathtt{a?}\}\}$ *and* $E'' = \{e \in E_\hbar \mid \vdash_\hbar e \wedge \ell_\hbar(e) \in \mathcal{P}_{\mathcal{U}} \times \mathtt{A}^!\}$*. Let* $\sharp$ *be the relation on* $(E_1 \times E_2) \cup (E_2 \times E_1)$ *of Definition A.10 calculated on* $\mathcal{E}_1 = \langle E_1, \#_1, \vdash'_1, \ell_1 \rangle$ *and* $\mathcal{E}_2 = \langle E_2, \#_2, \vdash'_2, \ell_2 \rangle$ *where* $(X \setminus \tilde{E}, e) \in \vdash'_i$ *if* $(X, e) \in \vdash_i$*, with* $\tilde{E} = \{e \in E_\hbar \mid \vdash_\hbar e\}$*.*

We define their buffered turn *composition* $\mathcal{E}_1 \tilde{\boxdot}^{\hbar}_{\{\mathtt{a?}\}} \mathcal{E}_2$ *as the* ES $\langle E, \#, \vdash, \ell \rangle$ *where*

$$
\begin{array}{rcl}
E & = & (E_1 \cup E_2) \setminus E'' \\
\# & = & (\#_1 \cup \#_2 \cup \sharp) \cap (E \times E) \\
\ell & = & (\ell_1 \cup \ell_2)|_E
\end{array}
$$

and $\vdash$ *is obtained saturating the following relation*

$$
\begin{array}{rll}
\vdash' = & \{(\emptyset, e) \mid e \in E' \ \vee \ \ell(e) \in \mathcal{P}_{\mathcal{U}} \times \{\checkmark\}\} & \\
& \cup \{(\{e\}, e') \mid e \in E' \ \wedge \ (\emptyset, e') \in \vdash_i \ \wedge \ \ell(e) \in \ \mathcal{P}_{\mathcal{U}} \times (\mathtt{A}^! \cup \{\checkmark\}) \ \wedge \ i \neq \hbar\} & (a) \\
& \cup \{(X \cup Y, e) \mid X \neq \emptyset \ \wedge \ X \vdash_i e \ \wedge \ \ell(e) \in \mathcal{P}_{\mathcal{U}} \times (\mathtt{A}^! \cup \{\checkmark\}) & \\
& \qquad \wedge \exists \hat{e} \in E_\hbar \setminus E'', \ Y \subseteq E_\hbar \setminus E''. \ Y \vdash_\hbar \hat{e} & (b) \\
& \qquad \wedge \ E' \cap Y \neq \emptyset \ \wedge \ X \dagger (Y \setminus E') \ \wedge \ X \cup Y \in \mathit{Con} \wedge \ i \neq \hbar\} & \\
& \cup \{(X \cup Y, e) \mid Y \neq \emptyset \ \wedge \ Y \subseteq E_\hbar \setminus E'' \ \wedge \ Y \vdash_\hbar e \wedge \ \ell(e) \in \mathcal{P}_{\mathcal{U}} \times (\mathtt{A}^! \cup \{\checkmark\}) & \\
& \qquad \wedge \exists \hat{e} \in E_i. \ X \vdash_i e \ \wedge \ E' \cap Y \neq \emptyset & (c) \\
& \qquad \wedge \ X \dagger (Y \setminus E') \ \wedge \ X \cup Y \in \mathit{Con} \wedge \ i \neq \hbar\} & \\
& \cup \{(X \cup Y \cup \{e'\}, e) \mid X \neq \emptyset \ \wedge \ X \vdash_i e \ \wedge \ \ell(e) \in \mathcal{P}_{\mathcal{U}} \times \mathtt{A}^? & \\
& \qquad \wedge \exists \hat{e} \in E_\hbar \setminus E'', \ Y \subseteq E_\hbar \setminus E''. \ Y \vdash_\hbar \hat{e} & \\
& \qquad \wedge \ E' \cap Y \neq \emptyset \ \wedge \ X \dagger (Y \setminus E') \ \wedge \ X \cup Y \cup \{e'\} \in \mathit{Con} & (d) \\
& \qquad \wedge \ \ell(e') = \mathit{co}(\ell(e)) \ \wedge \ e' \notin X \cup Y \wedge \ i \neq \hbar\} & \\
& \cup \{(X \cup Y \cup \{e'\}, e) \mid Y \neq \emptyset \ \wedge \ Y \subseteq E_\hbar \setminus E'' \ \wedge \ Y \vdash_\hbar e \wedge \ \ell(e) \in \mathcal{P}_{\mathcal{U}} \times \mathtt{A}^? & \\
& \qquad \wedge \exists \hat{e} \in E_i. \ X \vdash_i \hat{e} & \\
& \qquad \wedge \ E' \cap Y \neq \emptyset \ \wedge \ X \dagger (Y \setminus E') \ \wedge \ X \cup Y \cup \{e'\} \in \mathit{Con} & (e) \\
& \qquad \wedge \ \ell(e') = \mathit{co}(\ell(e)) \ \wedge \ e' \notin X \cup Y \wedge \ i \neq \hbar\} &
\end{array}
$$

where $i, j \in \{1, 2\}$ *and* $i \neq j$.

We pinpoint the main difference among the turn composition and the turn buffered composition. Assume that one of the two session types may perform an internal choice firing the event e. The enablings of the event structure $\mathcal{E}$, which is the turn composition of the $\mathcal{E}_1$ and $\mathcal{E}_2$ denoting the two session types, contains the $\vdash e$ and the execution of the event must trigger the execution of the matching external choice in the other session type (condition (a) of $\vdash'$ in Definition A.12). Furthermore all the other enabled events are in conflict with the chosen one, either because these events are guarding other branches in the same session type, ore because they are internal choices in the other session type. The remainder of this event structure ($\mathcal{E}[e]$) should perform the corresponding external choice. Assume that this correspond to the event e'. We must have that $\mathcal{E}[e][e']$ allows again just internal choice. This is captured by condition (a) of Definition A.14. We have then to calculate all the other enablings, that obviously depend on the internal choice done before (the buffered turn composition depends on the name of the external action to perform) and on the side where this action has to be done. The cases (b) – (e) of the above definition just correspond to the two cases of the previous one, thus cases (c) and (d) are the one corresponding to the case (b) of Definition A.12 and they take into account the side where the event representing the external choice has to be done.

Example A.15. *Consider the event structure* $\mathcal{E}_{\mathsf{B}}$ *of Example 5.21 and* $\mathcal{E}'_{\mathsf{A}} = (E'_{\mathsf{A}}, \#'_{\mathsf{A}}, \vdash'_{\mathsf{A}}, \ell'_{\mathsf{A}})$*, where* $E'_{\mathsf{A}} = \{e_7, e_9\}$*,* $\#'_{\mathsf{A}} = \emptyset$*,* $\vdash'_{\mathsf{A}} = \{\vdash'_{\mathsf{A}} e_7, e_7 \vdash'_{\mathsf{A}} e_9\}$*,* $\ell'_{\mathsf{A}}(e_7) = \mathsf{A} : \mathtt{b!}$ *and* $\ell'_{\mathsf{A}}(e_9) = \mathsf{A} : \checkmark$.

Take $\mathtt{b?} \in \mathtt{A}$*, then* $E' = \{e_8\}$ *and* $\iota = 2$*. The buffered turn composition* $\mathcal{E}'_{\mathsf{A}} \tilde{\boxdot}^2_{\{\mathtt{b?}\}} \mathcal{E}_{\mathsf{B}}$ *has the following set of events:* $\{e_2, e_4, e_6, e_7, e_8, e_9, e_{10}, e_{12}\}$*, as* E'' *is empty. The relation* $\sharp$ *is empty as well, hence the conflict relation is just the union of the two conflict relations. The enablings are:*

$$
\vdash \ = \ \{\vdash e_8, e_8 \vdash e_7, \{e_7, e_8\} \vdash e_{10}, \{e_7, e_8, e_{10}\} \vdash e_9, \{e_7, e_8, e_{10}\} \vdash e_{12}\}
$$

where $\vdash e_8$ *and* $e_8 \vdash e_7$ *are obtained with the clause (a) of Definition A.14,* $\{e_7, e_8\} \vdash e_{10}$ *and* $\{e_7, e_8, e_{10}\} \vdash e_{12}$ *derive from clause (d) of Definition A.14 (*$\{e_7, e_8\} \vdash e_{10}$ *because of* $e_8 \vdash_{\mathsf{B}} e_{10}$*,* $e_7 \vdash'_{\mathsf{A}} e_9$ *and* $\{e_7\} \dagger$

$\{e_8\}$, *and similarly for* $\{e_7, e_8, e_{10}\} \vdash e_{12}$*), and finally* $\{e_7, e_8, e_{10}\} \vdash e_9$ *is obtained using the clause (e) of Definition A.14 (because of* $e_7 \vdash_{\mathsf{A}} e_9$*,* $e_8 \vdash_{\mathsf{B}} e_{10}$*,* $\{e_7\} \dagger \{e_8\}$ *and* $\{e_7, e_8, e_{10}\} \in Con$*).*

A.4. Denotational semantics of session types

Consider the denotational semantics of session types (Definition 5.19). As already noticed, session types have recursion hence the standard machinery on fixed points is needed. Two event structures can be put in an ordering relation as follows. Intuitively, when $\mathcal{E} \trianglelefteq \mathcal{E}'$ then (i) each configuration of $\mathcal{E}$ is also a configuration of $\mathcal{E}'$, and (ii) each configuration of $\mathcal{E}'$ where the events are those of $\mathcal{E}$, is a configuration of $\mathcal{E}$ as well. The resulting relation is a partial order of ESs, and each ω-chain of ESs has a least upper bound [11].

Definition A.16 (Ordering of ESs [11]). *Let* $\mathcal{E} = (E, \#, \vdash, \ell)$ *and* $\mathcal{E}' = (E', \#', \vdash', \ell')$ *be two* ES*s. Then we write* $\mathcal{E} \trianglelefteq \mathcal{E}'$ *whenever:*

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

By putting the least upper bound of ω-chain of ESs

$$\mathcal{E}_1 \trianglelefteq \mathcal{E}_2 \trianglelefteq \cdots \trianglelefteq \mathcal{E}_n \trianglelefteq \ldots$$

as $\bigsqcup \mathcal{E}_i = (\bigcup_i E_i, \bigcup_i \#_i, sat(\bigcup_i \vdash_i), \bigcup_i \ell_i)$ it suffices to say that ESs are a *complete partial order*. This, together with the fact that certain operations are continuous (in our case lifting and sum), guarantees the existence of fixed points. 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 *fix* $\Gamma = \bigsqcup \mathbf{F}(\underline{\emptyset})$. It is standard to prove that the operators used by the denotational semantics in Figure 6, *i.e.* sum and lifting, are continuous.

With respect to the semantics presented in Figure 6, when considering turn-based configurations, we have to add the denotation of $\mathbf{0}$, which is obviously $[\![\mathbf{0}]\!]^{\mathsf{A}}_{\rho} = \langle \emptyset, \emptyset, \emptyset, \emptyset \rangle$. The Figure 8 contains the whole denotational semantics of turn-based configurations.

$$(\!|P_1 \parallel P_2|\!)^{\mathsf{AB}} = \quad [\![P_1]\!]^{\mathsf{A}}_{\rho} \boxdot [\![P_2]\!]^{\mathsf{B}}_{\rho} \text{ where } [\![P_i]\!]^{\mathsf{A}_i}_{\rho} = (E_i,\ \#_i,\ \vdash_i,\ \ell_i) \text{ are such that } E_1 \cap E_2 = \emptyset$$

$$(\!|[\mathtt{a!}]P_1 \parallel P_2|\!)^{\mathsf{AB}} = \quad [\![P_1]\!]^{\mathsf{A}}_{\rho} \tilde{\boxdot}^2_{\{\mathtt{a?}\}} [\![P_2]\!]^{\mathsf{B}}_{\rho} \text{ where } [\![P_i]\!]^{\mathsf{A}_i}_{\rho} = (E_i,\ \#_i,\ \vdash_i,\ \ell_i) \text{ are such that } E_1 \cap E_2 = \emptyset$$

Figure 8: Denotational semantics of turn-based configurations (symmetric rules for B omitted).

Observe that the case of the presence in the configuration of the one-position buffer is treated separately, as in this case the composition of the two event structures has to obey to the conditions given in the turn buffered composition.

We observe that the events enabled at the initial configuration of an ES obtained by a session type are labelled uniformly:

Lemma A.17. *Let* P *be a session type, and let* $[\![P]\!]^{\mathsf{A}}_{\emptyset} = (E, \#, \vdash, \ell)$ *be the associated event structure. Let* $E' = \{e \in E \mid \vdash e\}$*. Then, either* $\ell(E') \subseteq \mathcal{P}_{\mathcal{U}} \times \mathtt{A^!}$ *or* $\ell(E') \subseteq \mathcal{P}_{\mathcal{U}} \times \mathtt{A^?}$ *or* $\ell(E') \subseteq \mathcal{P}_{\mathcal{U}} \times \{\checkmark\}$*.*

Proof. An easy inspection on how the ESs are defined in Figure 6. □

We can state precisely what are the kind of labels of the enabled events of an ES stemming from a turn-based configuration as well.

Lemma A.18. *Let S be a turn-based configuration, and let $(\!|S|\!)^{\mathsf{A}} = (E, \#, \vdash, \ell)$ be the associated event structure. Let $E' = \{e \in E \mid \vdash e\}$. If $E' \neq \emptyset$ then either $\ell(E') \subseteq \mathcal{P}_{\mathcal{U}} \times (\mathtt{A}^{!} \cup \{\checkmark\})$ or $\ell(E') \subseteq \mathcal{P}_{\mathcal{U}} \times (\mathtt{A}^{?} \cup \{\checkmark\})$.*

Proof. It follows by an easy inspection of the clause (a) of Definition A.12 and Definition A.14. □

We relate the moves in event structures denoting the turn-based configurations to the turn-style semantics.

Lemma A.19. *Let P and Q two session types. Consider $\mathcal{E} = (\!|P \parallel Q|\!)^{\mathsf{AB}}$. Assume that $\vdash e$ is an enabling of $\mathcal{E}$, $\ell(e) = \mathsf{A} : \mathtt{a!}$ and $e \in \mathbf{E}_{\mathsf{A}}$. Then there exist two session types P' and P'' such that P is $\mathtt{a!} \,.\, P' \oplus P''$ and $\mathcal{E}[e]$ is precisely $(\!|[a]P' \parallel Q|\!)^{\mathsf{AB}}$.*

Proof. Let P and Q two session types and consider $\mathcal{E} = (\!|P \parallel Q|\!)^{\mathsf{AB}}$. Assume that $\vdash e$ is an enabling of $\mathcal{E}$, $\ell(e) = \mathsf{A} : \mathtt{a!}$ and $e \in \mathbf{E}_{\mathsf{A}}$.

As $(\!|P \parallel Q|\!)^{\mathsf{AB}}$ is $(\!|P \parallel Q|\!)^{\mathsf{AB}} = [\![P]\!]^{\mathsf{A}}_{\emptyset} \odot_{E'} [\![Q]\!]^{\mathsf{B}}_{\emptyset}$, where E' contains certainly e, it is trivial to observe, using Lemma A.17, that indeed there exists a session types P' and P'' such that P is $\mathtt{a!} \,.\, P' \oplus P''$. Furthermore, the initial events in $[\![P'']\!]^{\mathsf{A}}$ have labels different from $\mathtt{a!}$ as P is a session type.

We prove that $\mathcal{E}[e]$ is $\mathcal{E}' = (\!|[\mathtt{a?}]P' \parallel Q|\!)^{\mathsf{AB}}$.

We distinguish three cases: Q is either $\mathtt{b!} \,.\, Q' \oplus Q''$ or $\mathbf{1}$ or $\mathtt{a?} \,.\, Q' \oplus Q''$.

- if Q is $\mathtt{b!} \,.\, Q' \oplus Q''$, then E' contains also events from $[\![Q]\!]^{\mathsf{B}}_{\emptyset}$ and these events are in conflict with e, hence are eliminated from $\mathcal{E}[e]$ (and these are the set E'' of Definition A.14).

  $\mathcal{E}[e]$ does not contain any enabling of the form $\vdash e'$, as in the definition of $\vdash'$ in Definition A.12, the set $\{(\{e\}, e') \mid e \in E' \ \wedge \ e' \in E \setminus E' \ \wedge \ \emptyset \vdash_i e' \ \wedge \ \ell_i(e') = co(\ell(e))\}$ (clause (a)) is obviously empty.

  But $\mathcal{E}[e]$ does not contain any enabling at all, as each enabling $Z \vdash e'$ of $\mathcal{E}$ contains an event in E' which is in conflict with e. Now it is easy to see that also $\mathcal{E}'$ does not contain any enabling. We have that $(\!|[\mathtt{a?}]P' \parallel Q|\!)^{\mathsf{AB}} = [\![P]\!]^{\mathsf{A}}_{\emptyset} \tilde{\odot}^{2}_{\{\mathtt{a?}\}} [\![Q]\!]^{\mathsf{B}}_{\emptyset}$. $[\![Q]\!]^{\mathsf{B}}_{\emptyset}$ does not contain any enabling $\vdash e'$ with $\ell_2(e') = \mathsf{B} : \mathtt{a!}$ thus also $\mathcal{E}'$ does not contains any enabling of the form $\vdash e'$ as E' is empty. Consider the set E'' of Definition A.14, any enabling in $[\![Q]\!]^{\mathsf{B}}_{\emptyset}$ contains an event in E'', hence the sets (b) – (e) of the $\vdash'$ of this definition are empty.

  For the conflicts, it is trivial to see that they are exactly those of $\mathcal{E}'$ as in this case are calculated using a $\vdash'_2$ based on the $\vdash_2$ of $[\![Q]\!]^{\mathsf{B}}_{\emptyset}$ where the events from E'' are deleted.

  Hence $\mathcal{E}[e] = \mathcal{E}'$.

- Q is $\mathbf{1}$. Then $(\!|P \parallel \mathbf{1}|\!)^{\mathsf{AB}}$ is $[\![P]\!]^{\mathsf{A}}_{\emptyset} \odot [\![\mathbf{1}]\!]^{\mathsf{B}}_{\emptyset}$ and this has just enablings of the form $\vdash e$ as only clause (a) of Definition A.12 can be applied. The remainder of this executing e labelled as $\mathsf{A} : \mathtt{a!}$ gives an ES where the only enabling is the one of $[\![\mathbf{1}]\!]^{\mathsf{B}}_{\emptyset}$, as defined by $[\![P']\!]^{\mathsf{A}}_{\emptyset} \tilde{\odot}^{2}_{\{\mathtt{a?}\}} [\![\mathbf{1}]\!]^{\mathsf{B}}_{\emptyset} = (\!|[\mathtt{a?}]P' \parallel \mathbf{1}|\!)^{\mathsf{AB}}$. The conflicts of this ES are clearly those of $[\![P']\!]^{\mathsf{A}}_{\emptyset}$.

  Hence $\mathcal{E}[e] = \mathcal{E}'$.

- Q is $\mathtt{a?} \,.\, Q' \oplus Q''$. In this case $E' \subseteq \mathbf{E}_{\mathsf{A}}$ and E'' of Definition A.14 is empty.

  Let $\mathcal{E}_1 = [\![P]\!]^{\mathsf{A}}_{\emptyset} = (E_1, \#_1, \vdash_1, \ell_1)$ and $\mathcal{E}_2 = [\![Q]\!]^{\mathsf{B}}_{\emptyset} = (E_2, \#_2, \vdash_2, \ell_2)$.

  The enablings of $\mathcal{E}[e]$ are those arising from $\{e\} \cup Z \vdash e'$ of $\mathcal{E}$. Wlog we consider the enablings $\{e\} \cup Z \vdash e'$ such that if $\{e\} \cup Z' \vdash e'$ and $Z' \subseteq Z$ then $Z = Z'$ (those of the $\vdash'$ of Definition A.12).

  We distinguish several cases, according to the clause of Definition A.12 used to obtain the enabling $\{e\} \cup Z \vdash e'$.

  – the clause (a) of Definition A.12 has been used, hence $\{e\} \vdash e'$ and $\ell(e) = co(\ell e')$. But $\vdash e'$ is generated by clause (a) of Definition A.14.

– the clause (b) of Definition A.12 has been used. We have two sub-cases, either $e' \in E_1$ or $e' \in E_2$. Obviously $\ell(e') \in \mathcal{P}_{\mathcal{U}} \times (\mathtt{A}^! \cup \{\checkmark\})$.

  Assume $e' \in E_1$. We know that $\{e\} \cup Z$ can be partitioned into $X = \{e\} \cup Z \cap E_1$ and $Y = \{e\} \cup Z \cap E_2$ such that $X \dagger Y$. We have also that $X \vdash_1 e'$ and we know that there is an event $\hat{e}$ such that $Y \vdash \hat{e}$. Furthermore $X \cup Y$ is consistent.

  Consider now $\mathcal{E}_1[e]$. This is precisely $[\![P']\!]^{\mathsf{A}}_{\emptyset}$ and $X \setminus \{e\} \vdash e'$ is an enabling of this ES. Now, as E'' is empty, there exists an $\hat{e}$ such that $Y \vdash_2 \hat{e}$, but it is easy to observe that, assuming that Y the event in E', that precisely $X \setminus \{e\} \dagger (Y \setminus E')$. But these are the conditions of the clause (b) of Definition A.14.

  Assume instead that $e' \in E_2$. We know that $\{e\} \cup Z$ can be partitioned into $X = \{e\} \cup Z \cap E_1$ and $Y = \{e\} \cup Z \cap E_2$ such that $X \dagger Y$. We have also that $Y \vdash_2 e'$ and we know that there is an event $\hat{e}$ such that $X \vdash \hat{e}$. Furthermore $X \cup Y$ is consistent.

  Consider now $\mathcal{E}_1[e]$. This is precisely $[\![P']\!]^{\mathsf{A}}_{\emptyset}$ and $X \setminus \{e\} \vdash \hat{e}$ is an enabling of this ES. Now, as E'' is empty, we know that $Y \vdash_2 e$, but it is easy to observe that, assuming that Y the event in E', that precisely $X \setminus \{e\} \dagger (Y \setminus E')$. But these are the conditions of the clause (c) of Definition A.14.

– the clause (c) of Definition A.12 has been used. We have two sub-cases, either $e' \in E_1$ or $e' \in E_2$. Obviously $\ell(e') \in \mathcal{P}_{\mathcal{U}} \times \mathtt{A}^?$.

  Assume $e' \in E_1$. We know that there exists an event $e'' \in \{e\} \cup Z$ such that $\ell(e'') = co(\ell(e'))$, $e'' \in E_2$ and $(\{e\} \cup Z) \setminus \{e''\}$ can be partitioned into $X = (\{e\} \cup Z) \setminus \{e''\} \cap E_1$ and $Y = (\{e\} \cup Z) \setminus \{e''\} \cap E_2$ such that $X \dagger Y$. We have also that $X \vdash_1 e'$ and we know that there is an event $\hat{e}$ such that $Y \vdash \hat{e}$. Furthermore $X \cup Y \cup \{e''\}$ is consistent.

  Consider now $\mathcal{E}_1[e]$. This is precisely $[\![P']\!]^{\mathsf{A}}_{\emptyset}$ and $X \setminus \{e\} \vdash e'$ is an enabling of this ES. Now, as E'' is empty, there exists an $\hat{e}$ such that $Y \vdash_2 \hat{e}$, but it is easy to observe that, assuming that Y the event in E', that precisely $X \setminus \{e\} \dagger (Y \setminus E')$. But these are the conditions of the clause (d) of Definition A.14.

  Assume instead that $e' \in E_2$. We know that there exists an event $e'' \in \{e\} \cup Z$ such that $\ell(e'') = co(\ell(e'))$, $e'' \in E_1$ and $(\{e\} \cup Z) \setminus \{e''\}$ can be partitioned into $X = (\{e\} \cup Z) \setminus \{e''\} \cap E_1$ and $Y = (\{e\} \cup Z) \setminus \{e''\} \cap E_2$ such that $X \dagger Y$. We have also that $Y \vdash_1 e'$ and we know that there is an event $\hat{e}$ such that $X \vdash \hat{e}$. Furthermore $X \cup Y \cup \{e''\}$ is consistent.

  Consider now $\mathcal{E}_1[e]$. This is precisely $[\![P']\!]^{\mathsf{A}}_{\emptyset}$ and $X \setminus \{e\} \vdash \hat{e}$ is an enabling of this ES. Now, as E'' is empty, there exists an $\hat{e}$ such that $Y \vdash_2 e'$, but it is easy to observe that, assuming that Y the event in E', that precisely $X \setminus \{e\} \dagger (Y \setminus E')$. But these are the conditions of the clause (e) of Definition A.14.

Finally we have to show that the conflict relations coincide. But this is obvious just inspecting how the turn conflict relation of Definition A.14 is obtained. In fact consider the $\sharp$ calculated on $\mathcal{E}_1$ and $\mathcal{E}_2$. It cannot contain any pair with e. Now consider the turn conflict relation calculated on the enablings of $\mathcal{E}_1$ and on those of $\mathcal{E}_2$ where the events enabled with the empty set have been removed. As the cardinality of the enabling sets is driving this definition the result is exactly the same. □

Lemma A.20. *Let P and Q two session types. Consider $\mathcal{E} = (\!|P \,\|\, Q|\!)^{\mathsf{AB}}$. Assume that $\vdash e$ is an enabling of $\mathcal{E}$, $\ell(e) = \mathsf{B} : \mathtt{a?}$ and $e \in \mathbf{E}_{\mathsf{B}}$. Then there exists session types Q' and Q'' such that Q is $\mathtt{a!}\,.\,Q' \oplus Q''$ and $\mathcal{E}[e]$ is precisely $(\!|P \,\|\, [\mathtt{a?}]Q'|\!)^{\mathsf{AB}}$.*

Proof. As the proof of Lemma A.19 □

Lemma A.21. *Let P and Q two session types, and let $\mathtt{a?} \in \mathtt{A}^?$ be an action. Consider $\mathcal{E} = (\!|[\mathtt{a?}]P \,\|\, Q|\!)^{\mathsf{AB}}$ and assume that it contains an event $e \in \mathbf{E}_{\mathsf{B}}$ such that $\vdash e$ and $\ell(e) = \mathsf{B} : \mathtt{a?}$. Then there exist session types Q' and Q'' such that Q is $\mathtt{a?}\,.\,Q' + Q''$ and $\mathcal{E}[e]$ is precisely $(\!|P \,\|\, Q'|\!)^{\mathsf{AB}}$.*

Proof. Let P and Q two session types, let $\mathtt{a?} \in \mathtt{A}^?$, and consider $\mathcal{E} = (\!|[\mathtt{a?}]P \,\|\, Q|\!)^{\mathsf{AB}}$. Assume that $\vdash e$ is an enabling of $\mathcal{E}$, $\ell(e) = \mathsf{B} : \mathtt{a?}$ and $e \in \mathbf{E}_{\mathsf{B}}$.

As $(\!|[\mathtt{a?}]P \parallel Q|\!)^{\mathsf{AB}} = [\![P]\!]^{\mathsf{A}}_{\emptyset} \tilde{\boxdot}^{2}_{\{e\}} [\![Q]\!]^{\mathsf{B}}_{\emptyset}$, it is trivial to observe that there are session types Q' and Q'' such that Q is $\mathtt{a?}\,.\,Q' + Q''$.

We prove that $\mathcal{E}[e]$ is $\mathcal{E}' = (\!|P \parallel Q'|\!)^{\mathsf{AB}}$. Let $\mathcal{E}_1 = [\![P]\!]^{\mathsf{A}}_{\emptyset}$ and $\mathcal{E}_2 = [\![\mathtt{a?}\,.\,Q' + Q'']\!]^{\mathsf{B}}_{\emptyset}$.

Consider the enabling $Z \vdash e'$ of $\mathcal{E}[e]$. This clearly descends from an enabling $\{e\} \cup Z \vdash e'$ of $\mathcal{E}$. As before we consider just those that are not obtained by saturation.

We check how $\{e\} \cup Z \vdash e'$ is obtained. If $Z = \emptyset$ then e' is any event in $\mathcal{E}_1$ such that $\vdash_1 e'$ and $\ell(e') \in \{\mathsf{A}\} \times \mathtt{A}^{!}$, and clearly $\vdash e'$ is an enabling of $\mathcal{E}'$. Assume then that $Z \neq \emptyset$.

We have several cases, as $\{e\} \cup Z \vdash e'$ can be obtained using clause (b) – (e) of Definition A.14.

- assume it is obtained using the clause (b). Then $e' \in E_1$ and $X = (\{e\} \cup Z) \cap E_1$ is such that $X \vdash_1 e'$. Moreover we know that there exists $\hat{e} \in E_2$ and $Y = (\{e\} \cup Z) \cap E_2$ is such that $Y \vdash_2 \hat{e}$. Now $e \in Y$ and we know that $X \dagger (Y \setminus \{e\})$. But $\mathcal{E}_2[e]$ contains $Y \setminus \{e\} \vdash \hat{e}$ and we have all the ingredients for clause (b) of Definition A.12.

- assume it is obtained using the clause (c). Then $e' \in E_2$. The reasoning is as above, the clause (b) of Definition A.12 is used in $\mathcal{E}'$.

- assume it is obtained using the clause (d). Then $e' \in E_1$ and there exists an event $e'' \in E_2$ such that $\ell(e'') = co(\ell(e'))$. $X = (\{e\} \cup Z) \cap E_1$ is such that $X \vdash_1 e'$. Moreover we know that there exists $\hat{e} \in E_2$ and $Y = ((\{e\} \cup Z) \cup \{e''\}) \cap E_2$ is such that $Y \vdash_2 \hat{e}$. Now $e \in Y$ and we know that $X \dagger (Y \setminus \{e\})$. But $\mathcal{E}_2[e]$ contains $Y \setminus \{e\} \vdash \hat{e}$ and we have all the ingredients for clause (c) of Definition A.12.

- assume it is obtained using the clause (e). Then $e' \in E_2$. The reasoning is as above, the clause (c) of Definition A.12 is used in $\mathcal{E}'$.

For the conflicts of $(\!|P \parallel Q'|\!)^{\mathsf{AB}}$, it is trivial to see that these are precisely those of $\mathcal{E}[e]$. □

Lemma A.22. *Let P and Q two session types, and let $\mathtt{a?} \in \mathtt{A}^{?}$ be an action. Consider $\mathcal{E} = (\!|P \parallel [\mathtt{a?}]Q|\!)^{\mathsf{AB}}$ and assume that it contains an event $e \in \mathbf{E}_{\mathsf{A}}$ such that $\vdash e$ and $\ell(e) = \mathsf{A} : \mathtt{a?}$. Then there exists session types P' and P'' such that Q is $\mathtt{a?}\,.\,P' + P''$ and $\mathcal{E}[e]$ is precisely $(\!|P' \parallel Q|\!)^{\mathsf{AB}}$.*

Proof. As the proof of Lemma A.21. □

We can now prove the Theorem 5.20.

A.5. Proof of Theorem 5.20

We have to show that $(P \parallel Q, \twoheadrightarrow)$ and $(\emptyset, \rightarrow^{\ell}_{(\!|P \parallel Q|\!)})$ are bisimilar. By Lemma A.5, it suffices to show that $(P \parallel Q, \twoheadrightarrow)$ and $((\!|P \parallel Q|\!), \rightarrow^{\ell})$ (where the latter is the LTS over ES) are bisimilar.

- Assume $S \xrightarrow{\mathsf{A} : x}\!\!\!\!\twoheadrightarrow S'$, with $x \in \mathtt{A}^{!} \cup \mathtt{A}^{?} \cup \{\checkmark\}$ and $S \sim (\!|S|\!)^{\mathsf{AB}}$.

  Consider $x \in \mathtt{A}^{!}$. Hence S must be $\mathtt{a!}\,.\,P' \oplus P'' \parallel Q$. Assume then x is $\mathtt{a!}$.

  Consider now $(\!|\mathtt{a!}\,.\,P' \oplus P'' \parallel Q|\!)^{\mathsf{AB}}$. This is the event structure $\mathcal{E} = [\![\mathtt{a!}\,.\,P' \oplus P'']\!]^{\mathsf{A}}_{\emptyset} \boxdot_{E'} [\![Q]\!]^{\mathsf{B}}_{\emptyset}$ where $[\![\mathtt{a!}\,.\,P' \oplus P'']\!]^{\mathsf{A}_i}_{\emptyset} = (E_1,\ \#_1,\ \vdash_1,\ \ell_1)$, $[\![Q]\!]^{\mathsf{B}}_{\emptyset} = (E_2,\ \#_2,\ \vdash_2,\ \ell_2)$ and E' is the subset of $E_1 \cup E_2$ defined as follows: $\{e \in E_1 \cup E_2 \mid \vdash_i e \ \wedge\ \ell_i(e) \in \mathcal{P}_{\mathcal{U}} \times (\mathtt{A}^{!} \cup \{\checkmark\})\}$. Clearly there exists $e \in E'$ with $\ell_1(e) = \mathsf{A} : \mathtt{a!}$ and $\vdash_1 e$ is in $(E_1,\ \#_1,\ \vdash_1,\ \ell_1)$. By Definition A.12 $\vdash e$ is an enabling of $(\!|\mathtt{a!}\,.\,P' \oplus P'' \parallel Q|\!)^{\mathsf{AB}}$ hence $\mathcal{E} \xrightarrow{\ell(e)} \mathcal{E}[e]$.

  We show that $\mathcal{E}[e] = (\!|[\mathtt{a?}]P' \parallel Q|\!)^{\mathsf{AB}}$. There are the following cases, according to the form of Q:

  - $Q = \mathbf{1}$. Then $(\!|\mathtt{a!}\,.\,P' \oplus P'' \parallel \mathbf{1}|\!)^{\mathsf{AB}}$ contains only the enablings of the form $\emptyset \vdash e'$ with $e' \in E'$. The resulting event structure after executing e has just the enabling $\vdash e'$ with $\ell(e') = \mathsf{B} : \checkmark$ and the conflict are those arising from $[\![P']\!]^{\mathsf{A}}_{\emptyset}$. But this ES is precisely $(\!|[\mathtt{a?}]P' \parallel \mathbf{1}|\!)^{\mathsf{AB}} = [\![P']\!]^{\mathsf{A}}_{\emptyset} \tilde{\boxdot}^{2}_{\{\ell(e)\}} [\![\mathbf{1}]\!]^{\mathsf{B}}_{\emptyset}$ as $\{e' \in E_2 \mid \vdash_2 e' \ \wedge\ \ell(e') = \ell(e)\}$ is the empty set.

– $Q = \mathtt{b!}\,.\,Q' \oplus Q''$. Then the events in E' of $\mathcal{E} = [\![\mathtt{a!}\,.\,P' \oplus P'']\!]^{\mathsf{A}}_{\emptyset} \boxdot [\![Q]\!]^{\mathsf{B}}_{\emptyset}$ are the starting ones of $[\![\mathtt{a!}\,.\,P' \oplus P'']\!]^{\mathsf{A}}_{\emptyset}$ and of $[\![\mathtt{b!}\,.\,Q' \oplus Q'']\!]^{\mathsf{B}}_{\emptyset}$. The enablings are defined as in Definition A.12, and $\sharp$ contains also the pairs (e, e') with $e' \in E'$. Hence $\mathcal{E}[e]$ has no enabling (as each enabling $Z \vdash \hat{e}$ in $\mathcal{E}$ such that $Z \neq \emptyset$ contains an event in $E_2 \cap E'$) and the conflicts remaining are those arising from the events in $[\![P']\!]^{\mathsf{A}}_{\emptyset}$ and those in $[\![\mathtt{b!}\,.\,Q' \oplus Q'']\!]^{\mathsf{B}}_{\emptyset}$ calculated using definition A.10 with the enablings $Y \vdash_2 \hat{e}$ such that $\{e'\} \cup Y \vdash_2 \hat{e}$ is an enabling of $[\![\mathtt{b!}\,.\,Q' \oplus Q'']\!]^{\mathsf{B}}_{\emptyset}$, with $e' \in E'$. It is easy to see that this ES is precisely the one defined by $(\!|[\mathtt{a?}]P' \parallel Q|\!)^{\mathsf{AB}}$.
– $Q = \mathtt{a?}\,.\,Q' + Q''$. The events enabled in $\mathcal{E} = [\![\mathtt{a!}\,.\,P' \oplus P'']\!]^{\mathsf{A}}_{\emptyset} \boxdot [\![\mathtt{a?}\,.\,Q' + Q'']\!]^{\mathsf{B}}_{\emptyset}$ are those in E' and clearly $E' \subseteq E_1$.

  Consider $\mathcal{E}[e]$. The enablings of this event structure are those of the form $Z \vdash e'$ such that $\{e\} \cup Z \vdash e'$ is an enabling of $\mathcal{E}$. We show that these are precisely the enablings of $[\![P']\!]^{\mathsf{A}}_{\emptyset} \tilde{\boxdot}^2_{\{\mathtt{a?}\}} [\![\mathtt{a?}\,.\,Q' + Q'']\!]^{\mathsf{B}}_{\emptyset}$ and $E' = \{e \in E_2 \mid \vdash_2 e \ \wedge\ \ell_2(e) = \mathsf{B} : \mathtt{a?}\}$. Wlog we consider just the enablings $\{e\} \cup Z \vdash e'$ of $\mathcal{E}$ that are minimal (*i.e.* there is no enabling $Z' \vdash e'$ such that $Z' \subset \{e\} \cup Z$), as others enabling (not minimal) are obtained by this one by saturation. We have several cases:
  * if $Z = \emptyset$ then the only possibility is that $\ell(e') = \mathsf{B} : \mathtt{a?}$, but this is the unique enabling with this characteristic of $[\![P']\!]^{\mathsf{A}}_{\emptyset} \tilde{\boxdot}^2_{\{\mathtt{a?}\}} [\![\mathtt{a?}\,.\,Q' + Q'']\!]^{\mathsf{B}}_{\emptyset}$ according to condition (a) of Definition A.14.
  * $\ell(e') \in \{\mathsf{A}\} \times \mathtt{A}^{!}$ and $e' \in E_1$. We have that $((\{e\} \cup Z) \cap E_1) \dagger ((\{e\} \cup Z) \cap E_2)$ (as $\{e\} \cup Z \vdash e'$ is an enabling of $\mathcal{E}$) and, as $e' \in E_1$, there exists an event in $e'' \in Z \cap E_2$ such that $\ell_2(e'') = \ell_1(\mathtt{e})!$. We have that $(\{e\} \cup Z) \cap E_1 \vdash_1 e'$ and $Z \cap E_2 \vdash_2 \hat{e}$ for some $\hat{e} \in E_2$, as otherwise we would not have had $\{e\} \cup Z \vdash e'$, but this is precisely the clause (b) of the $\vdash$ in Definition A.14 applied to $[\![P']\!]^{\mathsf{A}}_{\emptyset}$ and $[\![\mathtt{a?}\,.\,Q' + Q'']\!]^{\mathsf{B}}_{\emptyset}$.
  * $\ell(e') \in \{\mathsf{B}\} \times \mathtt{A}^{!}$ and $e' \in E_2$. The reasoning is as above: We have that $((\{e\} \cup Z) \cap E_1) \dagger ((\{e\} \cup Z) \cap E_2)$ and then there exists an event in $e'' \in Z \cap E_2$ such that $\ell_2(e'') = \ell_1(\mathtt{e})!$. We have that $(\{e\} \cup Z) \cap E_1 \vdash_1 \hat{e}$ and $Z \cap E_2 \vdash_2 e'$ for some $\hat{e} \in E_1$, as otherwise we would not have had $\{e\} \cup Z \vdash e'$, and this is again the clause (c) of the $\vdash'$ in Definition A.14 applied to $[\![P']\!]^{\mathsf{A}}_{\emptyset}$ and $[\![\mathtt{a?}\,.\,Q' + Q'']\!]^{\mathsf{B}}_{\emptyset}$.
  * $\ell(e') \in \{\mathsf{A}\} \times \mathtt{A}^{?}$ and $e' \in E_1$. Then we know that there must be an event in $e'' \in Z$ such that $\ell_2(e'') = co(\ell_1(e'))$ and furthermore $(\{e\} \cup Z) \cap E_1 \vdash_1 e'$, $(Z \cap E_2) \setminus \{e''\} \vdash_2 \hat{e}$ for some $\hat{e} \in E_2$, and $((Z \cap E_2) \setminus \{e''\}) \dagger ((\{e\} \cup Z) \cap E_1)$, as otherwise $\{e\} \cup Z \vdash e'$ would not have been an enabling. But these are precisely the conditions required by the clause (d) of the relation $\vdash'$ in Definition A.14 when $[\![P']\!]^{\mathsf{A}}_{\emptyset}$ and $[\![Q]\!]^{\mathsf{B}}_{\emptyset}$ are considered.
  * $\ell(e') \in \{\mathsf{B}\} \times \mathtt{A}^{?}$ and $e' \in E_2$. Similar as above, using clause (e) of the relation $\vdash'$ in Definition A.14.

  It is routine to check that the conflicts are the correct ones, namely those of $[\![P']\!]^{\mathsf{A}}_{\emptyset} \tilde{\boxdot}^2_{\{\mathtt{a?}\}} [\![Q]\!]^{\mathsf{B}}_{\emptyset}$ and those of $(\!|P \parallel Q|\!)^{\mathsf{AB}}[e]$ coincide.

Summing up, if $\mathtt{a!}\,.\,P' \oplus P'' \parallel Q \xrightarrow{\mathsf{A}:\mathtt{a!}}\!\!\!\!\rightarrow\ [\mathtt{a?}]P' \parallel Q$ then $(\!|\mathtt{a!}\,.\,P' \oplus P'' \parallel Q|\!)^{\mathsf{AB}}[e] = (\!|[\mathtt{a?}]P' \parallel Q|\!)^{\mathsf{AB}}$, $(\!|\mathtt{a!}\,.\,P' \oplus P'' \parallel Q|\!)^{\mathsf{AB}} \xrightarrow{\ell(e)} (\!|\mathtt{a!}\,.\,P' \oplus P'' \parallel Q|\!)^{\mathsf{AB}}[e]$ and $[\mathtt{a?}]P' \parallel Q \ \sim\ (\!|[\mathtt{a?}]P' \parallel Q|\!)^{\mathsf{AB}}[e]$.

Consider $x = \checkmark$, then S is $\mathbf{1} \parallel \tilde{Q}$ and S' is $\mathbf{0} \parallel \tilde{Q}$.
$\tilde{Q}$ can be either a session type Q or $\mathbf{0}$ or $[\mathtt{a?}]Q$ for some Q session type and $\mathtt{a?} \in \mathtt{A}^{?}$.

– $\tilde{Q}$ is Q for some session type. The ES $\mathcal{E} = (\!|\mathbf{1} \parallel Q|\!)^{\mathsf{AB}}$ is $[\![\mathbf{1}]\!]^{\mathsf{A}}_{\emptyset} \boxdot [\![\tilde{Q}]\!]^{\mathsf{B}}_{\emptyset}$ where $[\![\mathbf{1}]\!]^{\mathsf{A}}_{\emptyset} = (\{e\}, \emptyset, \emptyset, \{(e, \checkmark)\})$ and $[\![Q]\!]^{\mathsf{B}}_{\emptyset} = (E_2, \#_2, \vdash_2, \ell_2)$ and certainly $e \in E'$. The enablings of this event structure are, according to Definition A.12, of the form $\vdash e'$ with $e' \in E'$ and the conflicts are those of $\#_2 \cup \{(e, e') \mid e, e' \in E'\}$. Thus $\mathcal{E}[e]$ is $(E'', \#'', \emptyset, \ell'')$ where $E'' = E_2 \setminus \{e' \in E_2 \mid \exists e' \in E_2.\ (e, e') \in \{(e, e') \mid e, e' \in E'\}\}$ and $\#'', \ell''$ are the restriction of $\#_2$ and ℓ_2 to the events in E''. But this is precisely the ES associated to $\mathbf{0} \parallel Q$, thus $\mathcal{E} \xrightarrow{\ell(e)} \mathcal{E}[e]$, $\mathcal{E}[e] = (\!|\mathbf{0} \parallel Q|\!)^{\mathsf{AB}}$ and $\mathbf{0} \parallel Q \ \sim\ (\!|\mathbf{0} \parallel Q|\!)^{\mathsf{AB}}$.

– $\tilde{Q}$ is $\mathbf{0}$. The ES $\mathcal{E} = (\!|\mathbf{1} \parallel \mathbf{0}|\!)^{\mathsf{AB}}$ is just $[\![\mathbf{1}]\!]^{\mathsf{A}}_{\emptyset} \boxdot [\![\mathbf{0}]\!]^{\mathsf{B}}_{\emptyset}$ which is $[\![\mathbf{1}]\!]^{\mathsf{A}}_{\emptyset}$ and the remainder is just the empty ES, as required

– $\tilde{Q}$ is $[\mathtt{a?}]Q$ for some Q session type and $\mathtt{a?} \in \mathtt{A}^{?}$. $(\!|\mathbf{1} \parallel []?Q|\!)^{\mathsf{AB}}$ is $\mathcal{E} = [\![\mathbf{1}]\!]^{\mathsf{A}}_{\emptyset} \tilde{\boxdot}^{1}_{\{\mathtt{a?}\}} [\![[\mathtt{a?}]Q]\!]^{\mathsf{B}}_{\emptyset}$ has just the enabling allowing to do e. Hence $\mathcal{E}[e]$ has no enabling like $[\![nil]\!]^{\mathsf{A}}_{\emptyset} \tilde{\boxdot}^{1}_{\{\mathtt{a?}\}} [\![[\mathtt{a?}]Q]\!]^{\mathsf{B}}_{\emptyset}$ which is $(\!|\mathbf{0} \parallel []?Q|\!)^{\mathsf{AB}} = \mathcal{E}'$

Hence $\mathcal{E} \xrightarrow{\ell(e)} \mathcal{E}[e]$ and $\mathcal{E}[e]$ is precisely $(\!|S'|\!)^{\mathsf{AB}}$.

Consider finally $x = \mathtt{a?}$. Then S is $\mathtt{a?}\,.\,P' + P'' \parallel [\mathtt{a?}]Q$. S' is obviously $P' \parallel Q$

The event structure associated to $\mathtt{a?}\,.\,P' + P'' \parallel [\mathtt{a?}]Q$ is $\mathcal{E} = [\![\mathtt{a?}\,.\,P' + P'']\!]^{\mathsf{A}}_{\emptyset} \tilde{\boxdot}^{1}_{\{\mathtt{a?}\}} [\![Q]\!]^{\mathsf{B}}_{\emptyset}$ and $E' \neq \emptyset$ contains just $e \in E_1$ such that $\ell(e) = \mathsf{A} : \mathtt{a?}$. Consider $\mathcal{E}[e]$. The enablings $Z \vdash e'$ of $\mathcal{E}[e]$ are derived by those of the form $\{e\} \cup Z \vdash e'$ in $\mathcal{E}$. According to Definition A.14 these enablings are precisely those arising from $[\![\mathtt{a?}\,.\,P']\!]^{\mathsf{A}}_{\emptyset} \boxdot [\![Q]\!]^{\mathsf{B}}_{\emptyset}$ as the enablings $\{e\} \cup Z \vdash e'$ are obtained from enablings in where the events involved are those of $[\![\mathtt{a?}\,.\,P']\!]^{\mathsf{A}}_{\emptyset}$. The same reasoning applies to the conflict relation, hence $\mathcal{E} \xrightarrow{\ell(e)} \mathcal{E}[e]$, $\mathcal{E}[e] = (\!|P' \parallel Q|\!)^{\mathsf{AB}}$. Furthermore $P' \parallel Q \;\sim\; (\!|P' \parallel Q|\!)^{\mathsf{AB}}$.

- The case $S \xrightarrow{\mathsf{B} : x} S'$, with $x \in \mathtt{A}^{!} \cup \mathtt{A}^{?} \cup \{\checkmark\}$ is the same as above

- Assume now that $(\!|S|\!)^{\mathsf{AB}} \xrightarrow{\ell(e)} (\!|S|\!)^{\mathsf{AB}}[e]$.
  Assume $\ell(e)$ is of the form $\mathsf{A} : x$ with $x \in \mathtt{A}^{!}$. S must be $\mathtt{a!}\,.\,P' \oplus P'' \parallel Q$ (by Lemma A.18) and assume that x is $\mathtt{a!}$

  – S is $\mathtt{a!}\,.\,P' \oplus P'' \parallel Q$ and by Lemma A.19 we then know that $(\!|S|\!)^{\mathsf{AB}}[e]$ is $(\!|[\mathtt{a?}]P' \parallel Q|\!)^{\mathsf{AB}}$, and clearly $\mathtt{a?}\,.\,P' \oplus P'' \parallel Q \xrightarrow{\mathsf{A} : \mathtt{a!}} [\mathtt{a?}]P' \parallel Q$ and $[\mathtt{a?}]P' \parallel Q \;\sim\; (\!|[\mathtt{a?}]P' \parallel Q|\!)^{\mathsf{AB}}$.

  If $\ell(e)$ is of the form $\mathsf{A} : x$ with $x \in \mathtt{A}^{?}$ then S must be $\mathtt{a?}\,.\,P' + P'' \parallel [\mathtt{a?}]Q$ (by Lemma A.18) and assume that x is $\mathtt{a?}$

  – S is $\mathtt{a?}\,.\,P' + P'' \parallel [\mathtt{a?}]Q$, by Lemma A.22 we then know that $(\!|\mathtt{a?}\,.\,P' + P'' \parallel [\mathtt{a?}]Q|\!)^{\mathsf{AB}}[e]$ is $(\!|P] \parallel Q|\!)^{\mathsf{AB}}$, and clearly $\mathtt{a?}\,.\,P' + P'' \parallel [\mathtt{a?}]Q \xrightarrow{\mathsf{A} : \mathtt{a?}} P' \parallel Q$ and $P' \parallel Q \;\sim\; (\!|P' \parallel Q|\!)^{\mathsf{AB}}$.

  If $\ell(e)$ is $\mathsf{A} : \checkmark$ then S is $\mathbf{1} \parallel \tilde{Q}$.
  Again we have various cases, depending on $\tilde{Q}$.

  – If $\tilde{Q}$ is Q for some session type Q then $(\!|\mathbf{1} \parallel Q|\!)^{\mathsf{AB}}[e]$ has only the enabling $\vdash e'$ of $[\![Q]\!]^{\mathsf{B}}_{\emptyset}$, if any and the conflicts are those arising from $[\![Q]\!]^{\mathsf{B}}_{\emptyset}$, as expected. But this is precisely $(\!|\mathbf{0} \parallel Q|\!)^{\mathsf{AB}}$ Thus clearly $\mathbf{1} \parallel Q \xrightarrow{\checkmark} \mathbf{0} \parallel Q$ and $\mathbf{0} \parallel Q \;\sim\; (\!|\mathbf{0} \parallel Q|\!)^{\mathsf{AB}}$.

  – If $\tilde{Q}$ is $\mathbf{0}$ then $(\!|\mathbf{1} \parallel \mathbf{0}|\!)^{\mathsf{AB}}[e]$ is the empty event structure. Thus clearly $\mathbf{1} \parallel \mathbf{0} \xrightarrow{\checkmark} \mathbf{0} \parallel \mathbf{0}$ and $\mathbf{0} \parallel \mathbf{0} \;\sim\; (\!|\mathbf{0} \parallel \mathbf{0}|\!)^{\mathsf{AB}}$.

  – If $\tilde{Q}$ is $[\mathtt{a?}]Q'$ for some Q' session type and $\mathtt{a?} \in \mathtt{A}^{?}$, then $\mathcal{E} = (\!|\mathbf{1} \parallel Q|\!)^{\mathsf{AB}}$ is $[\![\mathbf{1}]\!]^{\mathsf{A}}_{\emptyset} \tilde{\boxdot}^{1}_{\{\mathtt{a?}\}} [\![Q']\!]^{\mathsf{B}}_{\emptyset}$ which has as enabling just $\vdash e$ with $\ell(e) = \mathsf{A} : \checkmark$, and as conflicts those of $[\![Q']\!]^{\mathsf{B}}_{\emptyset}$, but this is just $[\![\mathbf{0}]\!]^{\mathsf{A}}_{\emptyset} \boxdot [\![Q']\!]^{\mathsf{B}}_{\emptyset}$, thus clearly $\mathbf{1} \parallel [\mathtt{a?}]Q' \xrightarrow{\checkmark} \mathbf{0} \parallel [\mathtt{a?}]Q'$ and $\mathbf{0} \parallel [\mathtt{a?}]Q' \;\sim\; (\!|\mathbf{0} \parallel [\mathtt{a?}]Q'|\!)^{\mathsf{AB}}$.

  The cases where $\ell(e)$ is of the form $\mathsf{B} : x$ are similar to those above (using Lemma A.19 and Lemma A.21). □