Choreography Synthesis as Contract Agreement

Julien Lange
University of Leicester, UK
jlange@le.ac.uk

Alceste Scalas
University of Cagliari, Italy
alceste.scalas@unica.it

We propose a formal model for distributed systems, where each participant advertises its requirements and obligations as behavioural *contracts*, and where multiparty sessions are started when a set of contracts allows to synthesise a *choreography*. Our framework is based on the CO_2 calculus for contract-oriented computing, and borrows concepts and results from the session type literature.

It supports sessions where the number of participants is not determined beforehand, and keeps CO_2's ability to rule out participants that are *culpable* if contracts are not fulfilled at runtime. We show that we have *progress* and *session fidelity* in CO_2, as a result of the *honesty* of participants — i.e., their ability to always adhere to their contracts.

1 Introduction

Distributed applications are nowadays omnipresent but even for seemingly simple cases, there is still a pressing need to make sure they do work as their designers intended. Indeed, such systems are difficult to design, verify, implement, deploy, and maintain. Besides the intrinsic issues due to the underlying execution model (concurrency, physical distribution, etc.), applications have to be designed within a strange paradox: they are made of components that, on the one hand, collaborate with each other and, on the other hand, may compete for resources, or for achieving conflicting goals. This paradox is especially relevant in inter-organisational service-oriented scenarios, where services may be deployed by different entities: even under common policies, the implementations may reflect diverging and changing requirements, up to the point of departing from the agreed specifications. This issue is reflected in standards such as [16], which includes runtime monitoring and logging to check that interactions in SOAs actually adhere to agreed policies and service descriptions.

Along the lines of [5, 4], we propose a formal model for distributed systems where *contracts* drive interactions: components advertise behavioural contracts; such contracts are used at runtime to establish multiparty *agreements*, and such agreements steer the behaviour of components. Therefore, contracts are not just a specification or a design mechanism anymore, rather they become a pivotal element of the execution model.

In this work, we combine two approaches: *session types* [11] and *contract-oriented computing* [5]. From the former, we adopt concepts, syntax and semantics — and in particular, the interplay between local behaviours and choreographies (i.e., between local types and global types) as a method for specifying and analysing the interactions of participants in a distributed system. However, in our framework we do not assume that a participant will necessarily always adhere to its specification, nor that a global description is available beforehand to validate the system. From the contract-oriented computing approach, we adopt CO_2 [4], a generic contract-oriented calculus where participants advertise their requirements and obligations through contracts, and interact with each other once *compliant* contracts have been found. Here, we tailor CO_2 to a multiparty model where contracts have the syntax of local types. We say that contracts $c_1, \ldots, c_n$ are compliant when, roughly, they can be used to synthesise a choreography — i.e., a global type whose projections are $c_1, \ldots, c_n$ themselves [15]. Once a set of compliant contracts has

To appear in EPTCS.

been found, a CO_2 session may be established, wherein the participants who advertised the contracts can interact. However, in line with what may happen in real life scenarios, the runtime behaviour of these participants may then depart from the contracts: the calculus allows to model these situations, and reason about them.

1.1 Contributions

Our framework models multiparty contractual agreements as "tangible" objects, i.e., choreographies. This allows us to rely on results and properties from the session type literature — in particular, the well-formedness of a choreography ensures that contractual agreements enjoy knowledge of choice, error/deadlock freedom, and progress. Furthermore, it allows us to easily check that some meta-level properties are satisfied at runtime, e.g., on the number of involved participants, whether or not the session may terminate, etc.

Our adaptation of CO_2 to a multiparty, choreography-based contract model preserves the properties of the original calculus. In particular, if a system gets stuck, it is possible to identify which participants violated their contracts.

We also discuss how the properties of a well-formed choreography are reflected in a context where participants can misbehave. We introduce global progress and session fidelity in CO_2, again inspired by analogous concepts in theories based on session types. We show that they hold in systems where all participants are *honest* (i.e., they always respect their contracts in any context) — even when a participant takes part in multiple sessions.

Synopsis. The rest of the paper is structured as follows. In the rest of this section, we introduce an example that we use to motivate and illustrate our framework. In Section 2, we introduce a multiparty contract model based on choreography synthesis. In Section 3, we present our version of CO_2 and highlight its main features. In Section 4, we define the notion of *honesty* and its practical importance in our contract-oriented scenario. In Section 5, we present our results, which link the notion of honesty to the progress and safety of a CO_2 system (due to lack of space, the proofs are relegated to an online appendix [14]). Finally, we discuss related work and conclude in Section 6.

1.2 A motivating example

In this section, we introduce a running example to illustrate our framework. We use $\mathtt{A}, \mathtt{B}, \ldots$ for participant names, and $\mathtt{a}, \mathtt{a}', \mathtt{b}, \ldots$ for participant variables, and use the colour blue to highlight contracts.

Consider the following distributed scenario: an online store $\mathtt{A}$ allows two buyers $\mathtt{b}_1$ and $\mathtt{b}_2$ to make a joint purchase through a simplified protocol: after they both request the same item, a quote is sent to $\mathtt{b}_1$, who is then expected to either place an order (order) or end the session (bye); the store also promises to notify $\mathtt{b}_2$ about whether the order was placed (ok) or cancelled (bye). $\mathtt{A}$'s behaviour is described by the following contract:

$$c_{\mathtt{A}} \;=\; \mathtt{b}_1?\mathsf{req};\mathtt{b}_2?\mathsf{req};\mathtt{b}_1!\mathsf{quote};\big(\mathtt{b}_1?\mathsf{order};\mathtt{b}_2!\mathsf{ok}+\mathtt{b}_1?\mathsf{bye};\mathtt{b}_2!\mathsf{bye}\big)$$

What kind of contracts would be compliant with $c_{\mathtt{A}}$? One answer consists in the following contracts, advertised by buyers $\mathtt{B}_1$ and $\mathtt{B}_2$.

$$\begin{aligned} c_{\mathtt{B}_1} &= \mathtt{a}!\mathsf{req};\mathtt{a}?\mathsf{quote};\big(\mathtt{b}_2'!\mathsf{ok};\mathtt{a}!\mathsf{order}\oplus\mathtt{b}_2'!\mathsf{bye};\mathtt{a}!\mathsf{bye}\big)\\ c_{\mathtt{B}_2} &= \mathtt{a}'!\mathsf{req};\big(\mathtt{b}_1'?\mathsf{ok};\mathtt{a}'?\mathsf{ok}+\mathtt{b}_1'?\mathsf{bye};\mathtt{a}'?\mathsf{bye}\big)\end{aligned}$$

Here, $\mathtt{B}_1$ promises to send the request to the store (a), wait for the quote, and then notify the other buyer (b'_2) before accepting or rejecting the store offer; symmetrically, $\mathtt{B}_2$'s contract sends the request to the store (a'), and then expects to receive the same notification (either ok or bye) from both the other buyer (b'_1) and the store itself. Each contract represents the local viewpoint of the participant who advertises it: $c_{\mathtt{A}}$ represents the local viewpoint of the store, and thus it does not (and indeed, it cannot) capture the communications between $\mathtt{B}_1$ and $\mathtt{B}_2$.

An agreement among $c_{\mathtt{A}}$, $c_{\mathtt{B}_1}$ and $c_{\mathtt{B}_2}$ may be found by replacing the participant variables in each contract with actual names, e.g., with substitutions $\{^{\mathtt{A}}/_{\mathtt{a},\mathtt{a}'}\}$, $\{^{\mathtt{B}_1}/_{\mathtt{b}_1,\mathtt{b}'_1}\}$ and $\{^{\mathtt{B}_2}/_{\mathtt{b}_2,\mathtt{b}'_2}\}$. Such an agreement is based on the existence of the following choreography (i.e., global type), which can be synthesised similarly to what is done in [15]:

$$\begin{array}{rcl}\mathcal{G}_{\mathtt{AB}_1\mathtt{B}_2} & = & \mathtt{B}_1 \rightarrow \mathtt{A} : \mathsf{req} ; \mathtt{B}_2 \rightarrow \mathtt{A} : \mathsf{req} ; \mathtt{A} \rightarrow \mathtt{B}_1 : \mathsf{quote} ; \\ & & (\mathtt{B}_1 \rightarrow \mathtt{B}_2 : \mathsf{ok} ; \mathtt{B}_1 \rightarrow \mathtt{A} : \mathsf{order} ; \mathtt{A} \rightarrow \mathtt{B}_2 : \mathsf{ok} \quad + \quad \mathtt{B}_1 \rightarrow \mathtt{B}_2 : \mathsf{bye} ; \mathtt{B}_1 \rightarrow \mathtt{A} : \mathsf{bye} ; \mathtt{A} \rightarrow \mathtt{B}_2 : \mathsf{bye})\end{array}$$

The ability to synthesise $\mathcal{G}_{\mathtt{AB}_1\mathtt{B}_2}$ guarantees that the global type is well-formed and projectable back to the initial contracts $c_{\mathtt{A}}$, $c_{\mathtt{B}_1}$ and $c_{\mathtt{B}_2}$ (with the substitutions above); this, in turn, guarantees progress and safety [15] of the contractual agreement.

However, in a realistic scenario, the existence of a contractual agreement among participants does not guarantee that progress and safety will also hold at runtime: in fact, a participant may advertise a contract promising some behaviour, and then fail to respect it — either maliciously or accidentally. Such failure may then cascade on other participants, e.g., if they remain stuck waiting for a promised message that is never sent.

This sort of situations can be modelled using the CO_2 calculus. A CO_2 system for the store-and-two-customers example may be implemented as follows:

$$S_1 \;=\; (x,y,z)\big(\mathtt{A}[\mathsf{tell}_{\mathtt{A}} \downarrow_x c_{\mathtt{A}} \,.\, \mathsf{fuse} \,.\, P_{\mathtt{A}}] \;\mid\; \mathtt{B}_1[\mathsf{tell}_{\mathtt{A}} \downarrow_y c_{\mathtt{B}_1} \,.\, P_{\mathtt{B}_1}] \;\mid\; \mathtt{B}_2[\mathsf{tell}_{\mathtt{A}} \downarrow_z c_{\mathtt{B}_2} \,.\, P_{\mathtt{B}_2}]\big)$$

Here, participant $\mathtt{A}$ advertises its contract $c_{\mathtt{A}}$ to itself via the primitive $\mathsf{tell}_{\mathtt{A}} \downarrow_x c_{\mathtt{A}}$, where x is used as a session handle for interacting with other participants. $\mathtt{B}_1$ and $\mathtt{B}_2$ advertise their respective contracts to $\mathtt{A}$ with a similar invocation.

In this example, $\mathtt{A}$ also plays the role of *contract broker*: once all contracts have been advertised, the fuse prefix can establish a new session, based on the fact that the global agreement $\mathcal{G}_{\mathtt{AB}_1\mathtt{B}_2}$ can be synthesised from $c_{\mathtt{A}}$, $c_{\mathtt{B}_1}$ and $c_{\mathtt{B}_2}$. This new session is shared among participants $\mathtt{A}$, $\mathtt{B}_1$ and $\mathtt{B}_2$.

At this point, the execution of the system (i.e., the continuation of processes $P_{\mathtt{A}}$, $P_{\mathtt{B}_1}$ and $P_{\mathtt{B}_2}$) is not required to respect the contracts. In fact, we will see that when the contracts are violated, the calculus allows for *culpable* participants to be always ruled out. Furthermore, we will discuss *honesty*, i.e., the guarantee that a participant will always fulfil its advertised contracts — even in contexts where other participants fail to fulfil theirs. When such a guarantee holds, the contractual progress and safety are also reflected in the runtime behaviour of the CO_2 system.

Other possible agreements. Our contract model allows for other scenarios. For instance, a participant $\mathtt{B}_{12}$ may impersonate both customers, and promise to always accept the store offer, by advertising the following contract:

$$c_{\mathtt{B}_{12}} \;=\; \mathsf{a}''!\mathsf{req}; \mathsf{a}''!\mathsf{req}; \mathsf{a}''?\mathsf{quote}; \mathsf{a}''!\mathsf{order}; \mathsf{a}''?\mathsf{ok}$$

where the request to the store (a'') is sent twice (i.e., once for each impersonated customer). In this case, if we combine $c_{\mathtt{A}}$ and $c_{\mathtt{B}_{12}}$ with substitutions $\{^{\mathtt{A}}/_{\mathtt{a}''}\}$, $\{^{\mathtt{B}_{12}}/_{\mathtt{b}_1,\mathtt{b}_2}\}$, we can find an agreement by synthesising the following global type:

$$\mathcal{G}_{\mathtt{AB}_{12}} \quad = \quad \mathtt{B}_{12} \to \mathtt{A} : \mathsf{req} ; \mathtt{B}_{12} \to \mathtt{A} : \mathsf{req} ; \mathtt{A} \to \mathtt{B}_{12} : \mathsf{quote} ; \mathtt{B}_{12} \to \mathtt{A} : \mathsf{order} ; \mathtt{A} \to \mathtt{B}_{12} : \mathsf{ok}$$

Similarly to the previous case, this scenario may be modelled with the following CO_2 system:

$$S_2 \;=\; (x,w)\big(\mathtt{A}[\mathsf{tell}_{\mathtt{A}} \downarrow_x c_{\mathtt{A}} . \mathsf{fuse} . P_{\mathtt{A}}] \;\mid\; \mathtt{B}_{12}[\mathsf{tell}_{\mathtt{A}} \downarrow_w c_{\mathtt{B}_{12}} . P_{\mathtt{B}_{12}}]\big)$$

where the fuse prefix can now create a session involving $\mathtt{A}$ and $\mathtt{B}_{12}$.

The participants in the CO_2 systems S_1 and S_2 may also be combined, so to obtain:

$$\begin{aligned} S_{12} \;=\; (x,y,z,w)\big(\; & \mathtt{A}[\mathsf{tell}_{\mathtt{A}} \downarrow_x c_{\mathtt{A}} . \mathsf{fuse} . P_{\mathtt{A}}] \;\mid\; \mathtt{B}_1[\mathsf{tell}_{\mathtt{A}} \downarrow_y c_{\mathtt{B}_1} . P_{\mathtt{B}_1}] \;\mid\; \mathtt{B}_2[\mathsf{tell}_{\mathtt{A}} \downarrow_z c_{\mathtt{B}_2} . P_{\mathtt{B}_2}] \\ & \mid\; \mathtt{B}_{12}[\mathsf{tell}_{\mathtt{A}} \downarrow_w c_{\mathtt{B}_{12}} . P_{\mathtt{B}_{12}}] \;\big) \end{aligned}$$

In this case, after all contracts have been advertised to $\mathtt{A}$, either a session corresponding to $\mathcal{G}_{\mathtt{AB}_1\mathtt{B}_2}$, or to $\mathcal{G}_{\mathtt{AB}_{12}}$ may take place, thus involving a different number of participants depending on which contracts are fused. In such cases, it makes sense to consider whether one of the agreements should take precedence over the other, and which criteria should drive this choice.

2 A Choreography-Based Contract Model

We introduce a contract model based on concepts and results from the session types literature. Individual contracts are expressed using the syntax of local session types; while contractual compliance is based on global types synthesis: a set of contracts is compliant if it is possible to synthesise a choreography from it, as described in [15]. For simplicity, we adopt syntax and semantics in the style of [8, 10]: we use participant names (instead of channels) for message exchange, i.e., we consider systems with just one channel between each pair of participants.

Syntax & Semantics. Let $\mathbb{P}$ and $\mathscr{P}$ be disjoint sets of, respectively, *participant names* (ranged over by $\mathtt{A}, \mathtt{B}, \ldots$) and *participant variables* (ranged over by $\mathtt{a}, \mathtt{b}, \ldots$). Let $\underline{\mathtt{a}}, \underline{\mathtt{b}}$ range over $\mathbb{P} \cup \mathscr{P}$. The syntax of contracts below is parametrised wrt *sorts* (ranged over by $\mathtt{e}$) which abstract data types (either simple or complex). We use the colour blue for single contracts and green for *systems* of contracts.

$$\begin{array}{rcccccccc} T, T' & ::= & T \mid T' & \mid & \mathtt{A}\langle c \rangle & \mid & (\mathtt{AB}) : \rho & \mid & \mathbf{0} \\ c, c' & ::= & \bigoplus_{i \in I} \underline{\mathtt{a}}_{\mathtt{i}} ! \mathtt{e}_{\mathtt{i}} ; c_i & \mid & \sum_{i \in I} \underline{\mathtt{a}} ? \mathtt{e}_{\mathtt{i}} ; c_i & \mid & \mu\, \mathbf{x}.c & \mid & \mathbf{x} \end{array}$$

A contract c may be either: (*i*) an internal choice $\oplus$, with the intuitive semantics that after sending the message $\mathtt{e}_i$ to participant $\underline{\mathtt{a}}_i$, behaviour c_i take places; (*ii*) an external choice $\sum$, saying that if a message of sort $\mathtt{e}_i$ is received from $\underline{\mathtt{a}}$, then behaviour c_i takes place; or (*iii*) a recursive behaviour. We assume that $\forall i \neq j \in I \colon (\underline{\mathtt{a}}_i, \mathtt{e}_i) \neq (\underline{\mathtt{a}}_j, \mathtt{e}_j)$ in internal choices, $\forall i \neq j \in I \colon \mathtt{e}_i \neq \mathtt{e}_j$ in external choices, and that $\oplus$ and $+$ are associative and commutative. We write $\mathrm{fv}(c)$ for the free participant variables in c.

A system of contracts T may be either: (*i*) a parallel composition of systems $T \mid T'$; or (*ii*) a *named* contract $\mathtt{A}\langle c \rangle$, saying that participant $\mathtt{A}$ promises to behave according to c; (*iii*) a *queue* $(\mathtt{AB}) : \rho$ of messages from $\mathtt{A}$ to $\mathtt{B}$. In a system T, we assume that there is at most one queue per pair of participants, (i.e., one channel per direction), and that participant names are pairwise distinct.

We consider systems of contracts as processes whose semantics is given by the following main reduction rules (see the online appendix [14] for the omitted ones):

$$\begin{array}{rcl} \mathtt{A}\langle \mathtt{B!e}; c_0 \oplus c_1 \rangle \mid (\mathtt{AB}) : \rho \mid T & \xrightarrow{\mathtt{A} \rightarrow \mathtt{B:e}} & \mathtt{A}\langle c_0 \rangle \mid (\mathtt{AB}) : \rho \cdot \mathtt{e} \mid T \\ \mathtt{A}\langle \mathtt{B?e}; c_0 + c_1 \rangle \mid (\mathtt{BA}) : \mathtt{e} \cdot \rho \mid T & \xrightarrow{\mathtt{A} \leftarrow \mathtt{B:e}} & \mathtt{A}\langle c_0 \rangle \mid (\mathtt{BA}) : \rho \mid T \end{array}$$

The first rule says that, after an internal choice, participant $\mathtt{A}$ puts a message $\mathtt{e}$ on its queue for participant $\mathtt{B}$. The second rule says that $\mathtt{A}$'s external choice can receive a message of the right sort from an input queue $\mathtt{BA}$. We write $T \xrightarrow{\mathtt{A} \leftrightharpoons \mathtt{B:e}} T'$ when either $T \xrightarrow{\mathtt{A} \rightarrow \mathtt{B:e}} T'$ or $T \xrightarrow{\mathtt{A} \leftarrow \mathtt{B:e}} T'$, and $\mathtt{Q}(T)$ for the parallel composition of the empty queues connecting all pairs of participants in T.

Example 2.1. *From the example in Section 1.2, consider the instantiated contracts of the store* $\mathtt{A}$ *and its customer* $\mathtt{B}_{12}$. *We illustrate the initial system, and how it progresses:*

$$\begin{array}{rllll} T_{\mathtt{AB}_{12}Q} & = & \mathtt{A}\langle c_{\mathtt{A}} \{^{\mathtt{A}}/_{\mathtt{a}''}\} \{^{\mathtt{B}_{12}}/_{\mathtt{b}_1,\mathtt{b}_2}\} \rangle & \mid \mathtt{B}_{12}\langle c_{\mathtt{B}_{12}} \{^{\mathtt{A}}/_{\mathtt{a}''}\} \{^{\mathtt{B}_{12}}/_{\mathtt{b}_1,\mathtt{b}_2}\} \rangle & \mid (\mathtt{AB}_{12}) : [] \mid (\mathtt{B}_{12}\mathtt{A}) : [] \\ & = & \mathtt{A}\langle \mathtt{B}_{12}\mathtt{?req}; \mathtt{B}_{12}\mathtt{?req}; \ldots \rangle & \mid \mathtt{B}_{12}\langle \mathtt{A!req}; \mathtt{A!req}; \ldots \rangle & \mid (\mathtt{AB}_{12}) : [] \mid (\mathtt{B}_{12}\mathtt{A}) : [] \\ & \xrightarrow{\mathtt{B}_{12} \rightarrow \mathtt{A:req}} & \mathtt{A}\langle \mathtt{B}_{12}\mathtt{?req}; \mathtt{B}_{12}\mathtt{?req}; \ldots \rangle & \mid \mathtt{B}_{12}\langle \mathtt{A!req}; \ldots \rangle & \mid (\mathtt{AB}_{12}) : [] \mid (\mathtt{B}_{12}\mathtt{A}) : \mathtt{req} \\ & \xrightarrow{\mathtt{A} \leftarrow \mathtt{B}_{12}\mathtt{:req}} & \mathtt{A}\langle \mathtt{B}_{12}\mathtt{?req}; \ldots \rangle & \mid \mathtt{B}_{12}\langle \mathtt{A!req}; \ldots \rangle & \mid (\mathtt{AB}_{12}) : [] \mid (\mathtt{B}_{12}\mathtt{A}) : [] \end{array}$$

Choreography Synthesis as Compliance. We briefly introduce the compliance relation that tells whether some contracts can be combined to describe a correct interaction. We reuse the main results from [15]: a typing system which assigns a (unique) global type to a set of local types. We say that a set of contracts (i.e., local types) is *compliant* if it can be assigned a choreography, i.e., a global type.

For simplicity, we use only a subset of the global types originally supported (we conjecture that extending this would not pose any difficulties). The main difference is that, in the style of [8, 10], we replace channels with participant names.

The syntax of global types is as follows:

$$\mathcal{G} \quad ::= \quad \mathtt{A} \rightarrow \mathtt{B} : \mathtt{e}; \mathcal{G} \quad \mid \quad \mathcal{G} + \mathcal{G}' \quad \mid \quad \mathcal{G} \mid \mathcal{G}' \quad \mid \quad \mu \mathbf{x}.\mathcal{G} \quad \mid \quad \mathbf{x} \quad \mid \quad \mathbf{0}$$

where the first production means that a participant $\mathtt{A}$ sends a message of sort $\mathtt{e}$ to $\mathtt{B}$, then interactions in $\mathcal{G}$ take place; $\mathcal{G} + \mathcal{G}'$ means that either interactions in $\mathcal{G}$, or in $\mathcal{G}'$ take place; $\mathcal{G} \mid \mathcal{G}'$ means that interactions in $\mathcal{G}$ and $\mathcal{G}'$ are executed concurrently; the rest of the productions are for recursive interactions, and end.

Similarly to the original synthesis, we use judgements of the form $\Gamma \vdash T \blacktriangleright \mathcal{G}$, where Γ is an environment to keep track of recursion variables, T is a system of contracts, and $\mathcal{G}$ is the global type assigned to T. We say that a system of contracts T *has global type* $\mathcal{G}$, if one can infer the judgement $\circ \vdash T \blacktriangleright \mathcal{G}$ from the rules in the online appendix [14] (simplified from [15]) where $\circ$ is the empty context Γ. Essentially, the synthesis rules allow to execute a set of contracts step-by-step, while keeping track of the structure of the interactions in a global type.

The main properties that we are interested in — and that are guaranteed by the synthesis — is that the inferred global type is (i) well-formed, and (ii) projectable back to the original contracts. Essentially, this means that each local type must be single-threaded, and that *knowledge of choice* is preserved — i.e., each choice is made by exactly one participant, and all the others are either made aware of the choice, or they have the same behaviour whatever choice is made.

Example 2.2. *Building up on the example from Section 1.2, we combine the contract of store* $\mathtt{A}$ *with those of customers* $\mathtt{B}_1$ *and* $\mathtt{B}_2$*, and we obtain the system:*

$$
\begin{array}{rcl}
T_{\mathtt{AB}_1\mathtt{B}_2} & = & \mathtt{A}\langle c_{\mathtt{A}}\,\{{}^{\mathtt{B}_1}/_{\mathtt{b}_1}\}\,\{{}^{\mathtt{B}_2}/_{\mathtt{b}_2}\}\rangle \mid \mathtt{B}_1\langle c_{\mathtt{B}_1}\,\{{}^{\mathtt{A}}/_{\mathtt{a}}\}\,\{{}^{\mathtt{B}_2}/_{\mathtt{b}'_2}\}\rangle \mid \mathtt{B}_2\langle c_{\mathtt{B}_2}\,\{{}^{\mathtt{A}}/_{\mathtt{a}'}\}\,\{{}^{\mathtt{B}_1}/_{\mathtt{b}'_1}\}\rangle \\
 & = & \mathtt{A}\langle \mathtt{B}_1?\mathsf{req};\mathtt{B}_2?\mathsf{req};\mathtt{B}_1!\mathsf{quote};(\mathtt{B}_1?\mathsf{order};\mathtt{B}_2!\mathsf{ok}+\mathtt{B}_1?\mathsf{bye};\mathtt{B}_2!\mathsf{bye})\rangle \\
 & & \quad \mid \mathtt{B}_1\langle \mathtt{A}!\mathsf{req};\mathtt{A}?\mathsf{quote};(\mathtt{B}_2!\mathsf{ok};\mathtt{A}!\mathsf{order}\oplus\mathtt{B}_2!\mathsf{bye};\mathtt{A}!\mathsf{bye})\rangle \\
 & & \quad \mid \mathtt{B}_2\langle \mathtt{A}!\mathsf{req};(\mathtt{B}_1?\mathsf{ok};\mathtt{A}?\mathsf{ok}+\mathtt{B}_1?\mathsf{bye};\mathtt{A}?\mathsf{bye})\rangle
\end{array}
$$

which can be assigned the following global type:

$$
\begin{array}{rcl}
\mathcal{G}_{\mathtt{AB}_1\mathtt{B}_2} & = & \mathtt{B}_1\to\mathtt{A}:\mathsf{req}\,;\mathtt{B}_2\to\mathtt{A}:\mathsf{req}\,;\mathtt{A}\to\mathtt{B}_1:\mathsf{quote}\,; \\
 & & (\mathtt{B}_1\to\mathtt{B}_2:\mathsf{ok}\,;\mathtt{B}_1\to\mathtt{A}:\mathsf{order}\,;\mathtt{A}\to\mathtt{B}_2:\mathsf{ok} \quad + \quad \mathtt{B}_1\to\mathtt{B}_2:\mathsf{bye}\,;\mathtt{B}_1\to\mathtt{A}:\mathsf{bye}\,;\mathtt{A}\to\mathtt{B}_2:\mathsf{bye})
\end{array}
$$

that is to say that $\circ\vdash T_{\mathtt{AB}_1\mathtt{B}_2} \blacktriangleright \mathcal{G}_{\mathtt{AB}_1\mathtt{B}_2}$ *holds. Instead, if we combine the store* $\mathtt{A}$ *with* $\mathtt{B}_{12}$ *we have*

$$
\begin{array}{rcl}
T_{\mathtt{AB}_{12}} & = & \mathtt{A}\langle c_{\mathtt{A}}\,\{{}^{\mathtt{B}_{12}}/_{\mathtt{b}_1,\mathtt{b}_2}\}\rangle \mid \mathtt{B}_{12}\langle c_{\mathtt{B}_{12}}\,\{{}^{\mathtt{A}}/_{\mathtt{a}''}\}\rangle \\
 & = & \mathtt{A}\langle \mathtt{B}_{12}?\mathsf{req};\mathtt{B}_{12}?\mathsf{req};\mathtt{B}_{12}!\mathsf{quote};(\mathtt{B}_{12}?\mathsf{order};\mathtt{B}_{12}!\mathsf{ok}+\mathtt{B}_{12}?\mathsf{bye};\mathtt{B}_{12}!\mathsf{bye})\rangle \\
 & & \quad \mid \mathtt{B}_{12}\langle \mathtt{A}!\mathsf{req};\mathtt{A}!\mathsf{req};\mathtt{A}?\mathsf{quote};\mathtt{A}!\mathsf{order};\mathtt{A}?\mathsf{ok}\rangle \\
 & & \\
\mathcal{G}_{\mathtt{AB}_{12}} & = & \mathtt{B}_{12}\to\mathtt{A}:\mathsf{req}\,;\mathtt{B}_{12}\to\mathtt{A}:\mathsf{req}\,;\mathtt{A}\to\mathtt{B}_{12}:\mathsf{quote}\,;\mathtt{B}_{12}\to\mathtt{A}:\mathsf{order}\,;\mathtt{A}\to\mathtt{B}_{12}:\mathsf{ok}
\end{array}
$$

and, again, the judgement $\circ\vdash T_{\mathtt{AB}_{12}} \blacktriangleright \mathcal{G}_{\mathtt{AB}_{12}}$ *holds.*

3 A Multiparty Version of $\mathrm{CO_2}$

We introduce a version of the $\mathrm{CO_2}$ calculus (for COntract-Oriented computing) [4] adapted to multiparty contracts and sessions. Let $\mathbb{S}$ and $\mathscr{S}$ be disjoint sets of, respectively, *session names* (ranged over by $s, s', \dots$) and *session variables* (ranged over by $x, y, z \dots$). Let $u, v, \dots$ range over $\mathbb{S}\cup\mathscr{S}$.

Syntax & Semantics. The syntax of $\mathrm{CO_2}$ is given by the following productions:

$$
\begin{array}{lrcl}
\text{Processes} & P, Q & ::= & \sum_{i\in I} p_i\,.\,P_i \;\mid\; P \mid Q \;\mid\; (\vec{u},\vec{\mathtt{a}})P \;\mid\; X(\vec{u},\vec{\mathtt{a}}) \;\mid\; \mathbf{0} \\
\text{Prefixes} & p & ::= & \tau \;\mid\; \mathsf{tell}_{\underline{\mathtt{a}}}\!\downarrow_u c \;\mid\; \mathsf{fuse} \;\mid\; \mathsf{do}^{u}_{\underline{\mathtt{a}}}\,\mathsf{e} \\
\text{Latent contracts} & K & ::= & \downarrow_u \mathtt{A}\ \mathit{says}\ c \;\mid\; K \mid K \\
\text{Systems} & S & ::= & \mathtt{A}[P] \;\mid\; \mathtt{A}[K] \;\mid\; s[T] \;\mid\; S \mid S \;\mid\; (\vec{u},\vec{\mathtt{a}})S \;\mid\; \mathbf{0}
\end{array}
$$

$\mathrm{CO_2}$ features CCS-style processes, equipped with branching $\sum$ (not to be confused with the choice operator used in contracts), parallel composition $\mid$, restrictions of session and participant variables, and named process invocation. The prefixes are for internal action (τ), contract advertisement ($\mathsf{tell}\!\downarrow$), session creation upon contractual agreement (fuse), and execution of contractual actions (do). A latent contract of the form $\downarrow_u \mathtt{A}\ \mathit{says}\ c$ represents the promise of participant $\mathtt{A}$ to fulfil c by executing do-actions on a session variable u. $\mathrm{CO_2}$ systems may be parallel compositions of processes $\mathtt{A}[P]$ (where $\mathtt{A}$ is the participant executing P), *latent* contracts $\mathtt{A}[K]$ (where $\mathtt{A}$ is the participant to which the contracts in K have been advertised), and established sessions $s[T]$ (where s is a session name, and T is a system of *stipulated* contracts as in Section 2). We assume well-formed systems where each participant $\mathtt{A}$ has at most one process $\mathtt{A}[P]$. Note that $\mathrm{CO_2}$ process and system productions allow to delimit both session names/variables ($\vec{u}$) and participant variables ($\vec{\mathtt{a}}$), but *not* participant names, which are considered public.

We give the main reduction rules for the semantics of CO$_2$ (where $+$ and $|$ are standard associative and commutative operators):

$$[\text{CO}_2\text{-}\textsc{Tell}] \quad \mathtt{A}[\mathsf{tell}_{\mathtt{B}} \downarrow_x c \,.\, P + P' \mid Q] \;\rightarrow\; \mathtt{A}[P \mid Q] \;\mid\; \mathtt{B}[\downarrow_x \mathtt{A}\ \mathit{says}\ c]$$

$$[\text{CO}_2\text{-}\textsc{Fuse}] \quad \frac{K \rhd^{\sigma}_{\pi} T \qquad \vec{\mathtt{a}} = \mathrm{dom}(\pi) \qquad \vec{u} = \mathrm{dom}(\sigma) \qquad \mathrm{img}(\sigma) = \{s\} \qquad s \text{ fresh}}{(\vec{u}, \vec{\mathtt{a}})\,\big(\mathtt{A}\big[\mathsf{fuse} \,.\, P + P' \mid Q\big] \;\mid\; \mathtt{A}[K] \;\mid\; S\big) \;\rightarrow\; (s)\,(\mathtt{A}[P \mid Q]\,\sigma\pi \;\mid\; s[T \mid \mathrm{Q}(T)] \;\mid\; S\sigma\pi)}$$

$$[\text{CO}_2\text{-}\textsc{Do}] \quad \frac{T \xrightarrow{\mathtt{A} \leftrightharpoons \mathtt{B}:\mathtt{e}} T'}{s[T] \;\mid\; \mathtt{A}\big[\mathsf{do}^{s}_{\mathtt{B}}\,\mathtt{e} \,.\, P + P' \mid Q\big] \;\rightarrow\; s\big[T'\big] \;\mid\; \mathtt{A}[P \mid Q]}$$

[CO$_2$-Tell] allows a participant A to advertise a contract c to B; as a result, a new *latent* contract is created, recording the fact that it was promised by A. [CO$_2$-Fuse] establishes a new session: the latent contracts held in $\mathtt{A}[K]$ are combined, and their participant variables substituted, in order to find an *agreement*, i.e., a T which satisfies the relation $K \rhd^{\sigma}_{\pi} T$ (see Definition 3.2 below). Provided an agreement is found, fresh session variable s and participants names are shared among the parties, via substitutions σ and π; within the session, the involved contracts become *stipulated* (as opposed to "latent", before the agreement). Rule [CO$_2$-Do] allows A to perform an input/output action e towards B on session s, provided that T permits it. The omitted CO$_2$ rules are standard, and they are listed in the online appendix [14]: they cover internal actions, parametric processes, parallel composition, and delimitations.

When needed, we label CO$_2$ system transitions: $S \xrightarrow{\mathtt{A}:\,p} S'$ means that S reduces to S' through a prefix p fired by participant A.

Example 3.1. *Consider the CO$_2$ system:*

$$S \;=\; \mathtt{A}[\mathsf{do}^{s}_{\mathtt{B}}\,\mathsf{int} + \mathsf{do}^{s}_{\mathtt{B}}\,\mathsf{bool}] \;\;\mid\;\; s[\mathtt{A}\langle\mathtt{B!int}\rangle \mid \mathtt{B}\langle\mathtt{A?int}\rangle \mid (\mathtt{AB}):[] \mid (\mathtt{BA}):[]] \;\;\mid\;\; \mathtt{B}[\mathsf{do}^{s}_{\mathtt{A}}\,\mathsf{int}]$$

Here, the CO$_2$ process of participant A *can perform an action towards* B *on session s, with either a message of sort* int *or* bool*. However,* A*'s contract in s only specifies that* A *should send a message of sort* int *to* B*: therefore, according to rule* [CO$_2$-Do]*, only the first branch of* A *may be chosen, and the system reduces as follows.*

$$\begin{array}{lllllll} S & \xrightarrow{\mathtt{A}:\ \mathsf{do}^{s}_{\mathtt{B}}\,\mathsf{int}} & \mathtt{A}[\mathbf{0}] & \mid & s[\mathtt{A}\langle\mathbf{0}\rangle \mid \mathtt{B}\langle\mathtt{A?int}\rangle \mid (\mathtt{AB}):\mathtt{int} \mid (\mathtt{BA}):[]] & \mid & \mathtt{B}[\mathsf{do}^{s}_{\mathtt{A}}\,\mathsf{int}] \\ & \xrightarrow{\mathtt{B}:\ \mathsf{do}^{s}_{\mathtt{A}}\,\mathsf{int}} & \mathtt{A}[\mathbf{0}] & \mid & s[\mathtt{A}\langle\mathbf{0}\rangle \mid \mathtt{B}\langle\mathbf{0}\rangle \mid (\mathtt{AB}):[] \mid (\mathtt{BA}):[]] & \mid & \mathtt{B}[\mathbf{0}] \end{array}$$

A main difference between our adaptation of CO$_2$ and the original presentation comes from the way we specify session establishment. We adopt the session agreement relation defined below.

Definition 3.2 (Agreement relation $K \rhd^{\sigma}_{\pi} T$)**.** *Let $K \equiv \big|_{i \in I} \downarrow_{x_i} \mathtt{A}_i$ says c_i, such that $\forall i \neq j \in I : \mathtt{A}_i \neq \mathtt{A}_j$, and let $\pi \colon \mathscr{P} \rightarrow \mathbb{P}$ and $\sigma \colon \mathscr{S} \rightarrow \mathbb{S}$ be two substitutions mapping participant variables to names, and session variables to names, respectively. Also, let $T \equiv \big|_{i \in I} \mathtt{A_i}\langle c_i \rangle \pi$. We define:*

$$K \rhd^{\sigma}_{\pi} T \iff \begin{array}{l} \mathrm{dom}(\sigma) = \bigcup_{i \in I} \{x_i\} \quad \wedge \quad \mathrm{dom}(\pi) = \bigcup_{i \in I} \mathrm{fv}(c_i) \\ \wedge \\ \forall i \in I \colon \forall \mathtt{a} \in \mathrm{fv}(c_i) \colon \pi(\mathtt{a}) \neq \mathtt{A}_i \quad \wedge \quad \exists \mathscr{G} \colon \circ \vdash T \blacktriangleright \mathscr{G} \end{array}$$

Intuitively, a system of stipulated contracts T is constructed from a set of latent contracts K, using a substitution π that maps all the participant variables in K to the participant names in K itself. If it is possible to synthesise a global type $\mathscr{G}$ out of T, then the relation holds, and a contractual agreement exists. The first two conditions, on σ and π, guarantee that all the session and participant variables are indeed instantiated. The third condition ensures that within a contract c_i, belonging to $\mathtt{A}_i$, no free participant variable in c_i is substituted by $\mathtt{A}_i$ itself. Note that due to the condition imposed on K, each participant may have at most one contract per session.

Example 3.3. *We now illustrate how Definition 3.2 works. Consider the following CO*$_2$ *system, with* $\mathtt{A}$*,* $\mathtt{B}_1$*,* $\mathtt{B}_2$ *from Section 1.2, and* $T_{\mathtt{AB}_1\mathtt{B}_2}$ *from Example 2.2:*

$$\begin{array}{rcl} S_1 & = & (x,y,z)\big(\mathtt{A}[\mathsf{tell}_{\mathtt{A}} \downarrow_x c_{\mathtt{A}} \,.\, \mathsf{fuse} \,.\, P_{\mathtt{A}}] \;\mid\; \mathtt{B}_1[\mathsf{tell}_{\mathtt{A}} \downarrow_y c_{\mathtt{B}_1} \,.\, P_{\mathtt{B}_1}] \;\mid\; \mathtt{B}_2[\mathsf{tell}_{\mathtt{A}} \downarrow_z c_{\mathtt{B}_2} \,.\, P_{\mathtt{B}_2}]\big) \\ \to\to\to & & (x,y,z)\big(\mathtt{A}[\mathsf{fuse} \,.\, P_{\mathtt{A}}] \;\mid\; \mathtt{A}[\downarrow_x \mathtt{A} \;\mathit{says}\; c_{\mathtt{A}} \mid \downarrow_y \mathtt{B}_1 \;\mathit{says}\; c_{\mathtt{B}_1} \mid \downarrow_z \mathtt{B}_2 \;\mathit{says}\; c_{\mathtt{B}_2}] \;\mid\; \mathtt{B}_1[P_{\mathtt{B}_1}] \;\mid\; \mathtt{B}_2[P_{\mathtt{B}_2}]\big) \\ \xrightarrow{\mathtt{A}:\,\mathsf{fuse}} (s)S_1' & = & (s)\big(\mathtt{A}[P_{\mathtt{A}}]\sigma\pi \;\mid\; s[T_{\mathtt{AB}_1\mathtt{B}_2} \mid \mathsf{Q}(T_{\mathtt{AB}_1\mathtt{B}_2})] \;\mid\; \mathtt{B}_1[P_{\mathtt{B}_1}]\sigma\pi \;\mid\; \mathtt{B}_2[P_{\mathtt{B}_2}]\sigma\pi\big) \end{array}$$

$$\textit{where } \sigma = \{{}^{s}/_{x,y,z}\} \textit{ and } \pi = \{{}^{\mathtt{A}}/_{\mathtt{a},\mathtt{a}'}, {}^{\mathtt{B}_1}/_{\mathtt{b}_1,\mathtt{b}_1'}, {}^{\mathtt{B}_2}/_{\mathtt{b}_2,\mathtt{b}_2'}\}$$

The initial system S_1 *is the one considered in Section 1.2, where all the participants are ready to advertise their respective contracts to the store* $\mathtt{A}$*, by using a* $\mathsf{tell}_{\mathtt{A}}\downarrow$*-primitive. This has the effect of creating corresponding latent contracts within* $\mathtt{A}$*. Once all the latent contracts are in a same location, they may be* fuse*d. In this case, given* σ *and* π *as above, Definition 3.2 is indeed applicable: the domains of* σ *and* π *comply with the definition's premises, and we already saw that a system consisting of* $c_{\mathtt{A}}$*,* $c_{\mathtt{B}_1}$ *and* $c_{\mathtt{B}_2}$ *may be assigned a global type. Hence, a new session* s *is created, based on the system of contracts* $T_{\mathtt{AB}_1\mathtt{B}_2}$*, plus the queues connecting all pairs of participants. The session variables of the latent contracts being* fuse*d (i.e.,* x *for participant* $\mathtt{A}$*,* y *for* $\mathtt{B}_1$*, and* z *for* $\mathtt{B}_2$*) are all substituted with the fresh session name* s *in the processes* $P_{\mathtt{A}}$*,* $P_{\mathtt{B}_1}$ *and* $P_{\mathtt{B}_2}$*, via* σ*. Similarly for participant variables which are substituted with participant names, via* π*.*

The CO$_2$ semantic rules are to be considered up-to a standard structural congruence relation $\equiv$ (cf. [14]): we just point out that $\mathtt{A}[K] \mid \mathtt{A}[K'] \equiv \mathtt{A}[K \mid K']$ allows to select a compliant subset from a group of latent contracts, before performing a fuse — thus adding flexibility to the synthesis of choreographies.

Example 3.4. *Consider the system:*

$$\ldots \mathtt{B}[\mathsf{fuse} \,.\, P \mid Q] \;\mid\; \mathtt{B}\big[\downarrow_x \mathtt{A}_1 \;\mathit{says}\; \mathsf{a!int} \mid \downarrow_y \mathtt{A}_2 \;\mathit{says}\; \mathsf{a'?int} \mid \downarrow_z \mathtt{A}_3 \;\mathit{says}\; \mathsf{b?bool}\big] \ldots$$

The fuse *prefix cannot be fired: no contract matches* $\mathtt{A}_3$*'s, and thus the three latent contracts cannot be assigned a global type. However, by rearranging the system with congruence* $\equiv$*, we have:*

$$\ldots \mathtt{B}[\mathsf{fuse} \,.\, P \mid Q] \;\mid\; \mathtt{B}\big[\downarrow_x \mathtt{A}_1 \;\mathit{says}\; \mathsf{a!int} \mid \downarrow_y \mathtt{A}_2 \;\mathit{says}\; \mathsf{a'?int}\big] \;\mid\; \mathtt{B}[\downarrow_z \mathtt{A}_3 \;\mathit{says}\; \mathsf{b?bool}] \ldots$$

It is now possible to synthesise a global type $\mathtt{A}_1 \to \mathtt{A}_2 : \mathsf{int}$*, and a session may be created for* $\mathtt{A}_1$ *and* $\mathtt{A}_2$*.* $\mathtt{A}_3$*'s latent contract may be* fuse*d later on.*

3.1 Flexibility of Session Establishment

We highlight the flexibility of our definition of contract agreement, together with the semantics of CO$_2$, by discussing three features: (*i*) contracts may use a mix of participant names and variables, (*ii*) different contracts may use common participant variables, and (*iii*) the definition of agreement may be easily extended.

Contracts with both participant names and variables. A may want to sell an item to a *specific* participant B, via *any* shipping company that provides a package tracking system. A's contract may be:

$$\mathsf{B!price}; \mathsf{B?ack}; \mathsf{a!request}; \mathsf{a?tracking}; \mathsf{B!tracking}$$

saying that the seller A must send a price to the buyer B; once B has acknowledged, A must send a shipping request to a shipper a — who must send back a tracking number, which is then forwarded to B. This contract may be fused only if B takes part in the session, while the role of shipper a may be played by any participant.

Contracts sharing participant variables. Consider the CO_2 process:

$$\ldots \mathtt{A}[\mathsf{tell}_{\mathtt{A}\downarrow x}(\mathsf{b!request}) \ldots X(\vec{z}, \mathtt{b})] \ldots \quad \text{where } X(\vec{z}, \mathtt{b}) := (y, \mathtt{b}')\mathsf{tell}_{\mathtt{A}\downarrow y}(\mathsf{b'!quote}; \ldots \mathsf{b!address}) \ldots X(\vec{z}, \mathtt{b})$$

Here, A advertises two contracts: the first one ($\mathsf{b!request}$) is used by A to find a shipping company, and the second ($\mathsf{b'!quote}; \ldots \mathsf{b!address}$) to sell items. The two contracts are linked by the common variable b: whenever the first one is fused, variable b is instantiated to a participant name, say B, which is also substituted in the second. This means that whenever a new selling session starts, B will also be involved as the receiver of the address message.

Possible extensions. The participants firing fuse-primitives are playing the role of brokers in our framework. Depending on their implementation, brokers may also have some obligations in the contracts they fuse, or they may want to enforce some general policy — therefore they may have additional requirements before agreeing to start a session. For instance, a broker may not want to start a session with too many participants as it may be too resource demanding (too many connections etc.). Another broker may want to start sessions that terminates after a limited number of interactions, because it has a short life expectancy, e.g., due to an approaching scheduled maintenance. Another kind of broker may precisely want to start sessions which do not terminate, e.g., if the broker is interested in resilient services.

Several variations of the fuse primitive are possible thanks to the fact that we base contract agreements on objects representing the overall choreography. We introduce $\mathsf{fuse}[n]$, a version of fuse that only fuses sessions where there are at least n participants; fuse_T, which has the additional constraint that no recursive behaviour is allowed in the synthesised choreography (therefore ensuring that the session will eventually terminate), and fuse_R, which only creates sessions when the synthesised choreography never terminates (i.e., it only consists of recursive behaviours).

The three extensions may be defined directly via small modifications of Definition 3.2:

- $\mathsf{fuse}[n]$: we add the condition $|\mathscr{P}(\mathscr{G})| \geq n$, where $\mathscr{P}(\mathscr{G})$ is the set of participants in $\mathscr{G}$;
- fuse_T: we add the condition that there should not be any recursion variable $\mathbf{x}$ in $\mathscr{G}$;
- fuse_R: we add the condition that $\mathbf{0}$ does not appear in $\mathscr{G}$.

This kind of properties must be checked for at global level because it cannot always be decided by looking at the individual contracts. For instance, a participant might exhibit a recursive behaviour in one of the branches of an external choice, while the participant it interacts with may always choose a branch that is not recursive. Note that none of these variations actually affect the results that follow, since the original fuse primitive is also blocking. The variations only restrict some of its applications. Further variations of fuse are sketched in Section 6, as future work.

4 The Problem of Honesty

In this section, we discuss and define the notion of *honesty* [4], i.e., the ability of a participant to always fulfil its contracts, in any context. Broadly speaking, in our contract-oriented setting, honesty is the counterpart of well-typedness in a session type setting: the static proof that a participant always honours its contracts provides guarantees about its runtime behaviour.

As seen in Example 3.1, each do prefix within the process of a participant, say $\mathtt{A}[P]$, is driven by the contract that A promised to abide by. In a sense, CO_2 is *culpability-driven*, according to Definition 4.1 below: when a participant is culpable, it has the duty of making the session progress according to its contract.

Definition 4.1 (Culpability)**.** *Let S be a CO_2 system with a session s, i.e., $S \equiv (\vec{u}, \vec{\mathsf{a}})\,(\mathtt{A}[P] \mid s[T] \mid S_1)$. We say that* A *is* culpable in S (at session s) *when there exist* B *and* e *such that $T \xrightarrow{\mathtt{A} \Leftarrow \mathtt{B}:\mathsf{e}}$.*

A culpable participant can overcome its status by firing its do prefixes, according to [CO_2-Do], until another participant becomes culpable or session s terminates. Hence, as long as a culpable participant A does not enable a do-prefix matching a contractual action, A will remain culpable. Note that when a participant is involved in multiple sessions, it may result culpable in more than one of them.

When a participant A is always able to fulfil its contractual actions (i.e., overcome its culpability), no matter what other participants do, then it is said to be *honest* (cf. Definition 4.9). This is a desirable property in a distributed contract-oriented scenario: a participant may be stuck in a culpable condition either due to "simple" bugs (cf. Example 4.7), or due to the unexpected (or malicious) behaviour of other participants (cf. Example 5.6). Therefore, before deploying a service, its developers might want to ensure that it will always be able exculpate itself.

Formally, as in [1], we base the definition of honesty on the relationship between the ready sets of a contract, and those of a CO_2 process. We call the former *contract ready sets*, and the latter *process ready sets*. The concept of contract ready sets is similar to [9, 4, 1], where only bilateral contracts are considered. Here, we adapt it to suit our multiparty contract model.

Definition 4.2 (Contract Ready Sets)**.** *The* ready sets of a contract c, *written* $\text{CRS}(c)$, *are:*

$$\text{CRS}(c) = \begin{cases} \text{CRS}(c') & \text{if } c = \mu\,\mathbf{x}.c' \\ \{\{(\mathtt{A}_i, \mathsf{e}_i)\} \mid i \in I\} & \text{if } c = \bigoplus_{i \in I} \mathtt{A}_i!\mathsf{e}_i; c_i \quad \text{and} \quad I \neq \varnothing \\ \{\{(\mathtt{A}, \mathsf{e}_i) \mid i \in I\}\} & \text{if } c = \sum_{i \in I} \mathtt{A}?\mathsf{e}_i; c_i \end{cases}$$

Intuitively, when a participant A is bound to a contract c, the ready sets of c tell which interactions A must be able to perform towards other participants. Each interaction has the form of a pair, consisting of a participant name and a message sort. The interactions offered by an external choice are all available at once, while those offered by an internal choice are mutually exclusive.

Example 4.3. *Consider the system of contracts $T_{\mathtt{AB}_1\mathtt{B}_2}$ from Example 2.2 — and in particular, the stipulated contracts therein, with substitution $\pi = \{{}^{\mathtt{A}}/_{\mathtt{a},\mathtt{a}'}, {}^{\mathtt{B}_1}/_{\mathtt{b}_1,\mathtt{b}_1'}, {}^{\mathtt{B}_2}/_{\mathtt{b}_2,\mathtt{b}_2'}\}$ from Example 3.3:*

$$\begin{array}{lllll} \tilde{c_{\mathtt{A}}} & = & c_{\mathtt{A}}\pi & = & \mathtt{B}_1?\mathsf{req}; \mathtt{B}_2?\mathsf{req}; \mathtt{B}_1!\mathsf{quote}; (\mathtt{B}_1?\mathsf{order}; \mathtt{B}_2!\mathsf{ok} + \mathtt{B}_1?\mathsf{bye}; \mathtt{B}_2!\mathsf{bye}) \\ \tilde{c_{\mathtt{B}_1}} & = & c_{\mathtt{B}_1}\pi & = & \mathtt{A}!\mathsf{req}; \mathtt{A}?\mathsf{quote}; (\mathtt{B}_2!\mathsf{ok}; \mathtt{A}!\mathsf{order} \oplus \mathtt{B}_2!\mathsf{bye}; \mathtt{A}!\mathsf{bye}) \\ \tilde{c_{\mathtt{B}_2}} & = & c_{\mathtt{B}_2}\pi & = & \mathtt{A}!\mathsf{req}; (\mathtt{B}_1?\mathsf{ok}; \mathtt{A}?\mathsf{ok} + \mathtt{B}_1?\mathsf{bye}; \mathtt{A}?\mathsf{bye}) \end{array}$$

We have $\text{CRS}(\tilde{c_{\mathtt{A}}}) = \{\{(\mathtt{B}_1, \mathsf{req})\}\}$: in other words, at this point of the contract, an interaction is expected between A *and* $\mathtt{B}_1$ *(since* A *is waiting for* req*), while no interaction is expected between* A *and* $\mathtt{B}_2$*.*

Let us now equip $T_{\mathtt{AB_1B_2}}$ with one queue between each pair of participants, and let it perform the req*uest exchange between* $\mathtt{B_1}$ *and* $\mathtt{A}$*, with the transitions:*

$$T_{\mathtt{AB_1B_2}} \mid \mathrm{Q}(T_{\mathtt{AB_1B_2}}) \xrightarrow{\mathtt{B_1 \to A:req}} \xrightarrow{\mathtt{A \leftarrow B_1:req}} T'_{\mathtt{AB_1B_2}} \mid \mathrm{Q}(T_{\mathtt{AB_1B_2}})$$

We have that $\tilde{c_{\mathtt{A}}}$ in $T'_{\mathtt{AB_1B_2}}$ is now reduced to:

$$\tilde{c_{\mathtt{A}}}' = \mathsf{B_2?req; B_1!quote; (B_1?order; B_2!ok + B_1?bye; B_2!bye)}$$

and thus we have $\mathrm{CRS}(\tilde{c_{\mathtt{A}}}') = \{\{(\mathtt{B_2}, \mathsf{req})\}\}$*, i.e.,* $\mathtt{A}$ *is now waiting for a* request *from* $\mathtt{B_2}$.

If we let the system reduce further, $\tilde{c_{\mathtt{A}}}'$ reaches its external choice:

$$\tilde{c_{\mathtt{A}}}'' = \mathsf{B_1?order; B_2!ok + B_1?bye; B_2!bye}$$

Now, the ready sets become $\mathrm{CRS}(\tilde{c_{\mathtt{A}}}'') = \{\{(\mathtt{B_1}, \mathsf{order}), (\mathtt{B_1}, \mathsf{bye})\}\}$*, i.e.,* $\mathtt{A}$ *must handle both answers from* $\mathtt{B_1}$*. Instead, when $\tilde{c_{\mathtt{B_1}}}$ reduces to its internal choice, we have:*

$$\tilde{c_{\mathtt{B_1}}}'' = \mathsf{B_2!ok; A!order \oplus B_2!bye; A!bye}$$

Thus, its ready sets become $\mathrm{CRS}(\tilde{c_{\mathtt{B_1}}}'') = \{\{(\mathtt{B_2}, \mathsf{ok})\}, \{(\mathtt{B_2}, \mathsf{bye})\}\}$*:* $\mathtt{B_1}$ *is free to choose either branch.*

Example 4.3 shows that, when a *contract* c of a principal $\mathtt{A}$ evolves within a system T, its ready sets change. Now we need to define the counterpart of contract ready sets for $\mathrm{CO_2}$ processes, i.e., the *process ready sets*. Again, we adapt the definition from [1] to our multiparty contract model.

Definition 4.4 (Process Ready Set)**.** *For all CO_2 systems S, all participants* $\mathtt{A}, \mathtt{B}$ *and sessions u, we define the set of pairs:*[1]

$$\mathrm{PRS}^u_{\mathtt{A}}(S) = \left\{(\mathtt{B}, \mathtt{e}) \mid \exists \vec{v}, \vec{\mathtt{a}}, P, P', Q, S' : S \equiv (\vec{v}, \vec{\mathtt{a}}) \left(\mathtt{A}\left[\mathsf{do}^u_{\mathtt{B}}\, \mathtt{e} \,.\, P + P' \mid Q\right] \mid S_0\right) \mid S_1 \;\wedge\; u \notin \vec{v}\right\}$$

Intuitively, Definition 4.4 says that the process ready set of $\mathtt{A}$ over a session u in a system S contains the interactions that $\mathtt{A}$ is immediately able to perform with other participants through its $\mathsf{do}^u_{\_}$ prefixes. Just as in contract ready sets, the interactions are represented by participant/sort pairs.

Next, we want to characterise a weaker notion of the process ready set, so it only takes into account the first actions *on a specific session* that a participant is ready to make.

Definition 4.5 (Weak Process Ready Set)**.** *We write* $S \xrightarrow{\neq(\mathtt{A}:\, \mathsf{do}^u)} S'$ *iff:*

$$\exists \mathtt{B}, p \colon S \xrightarrow{\mathtt{B}:\, p} S' \implies (\mathtt{A} \neq \mathtt{B} \quad \vee \quad \forall \mathtt{e} \colon \forall \mathtt{C} \colon p = \mathsf{do}^v_{\mathtt{C}}\, \mathtt{e} \implies u \neq v)$$

We then define the set of pairs $\mathrm{WPRS}^u_{\mathtt{A}}(S)$ *as:*

$$\mathrm{WPRS}^u_{\mathtt{A}}(S) = \left\{(\mathtt{B}, \mathtt{e}) \mid \exists S' \colon S \xrightarrow{\neq(\mathtt{A}:\, \mathsf{do}^u)}{}^{*} S' \wedge (\mathtt{B}, \mathtt{e}) \in \mathrm{PRS}^u_{\mathtt{A}}\left(S'\right)\right\}$$

In Definition 4.5, we are not interested in the actions that do not relate to the session u. Thus, we allow the system to evolve either by (*i*) letting any other participant other than $\mathtt{A}$ do an action, or (*ii*) letting $\mathtt{A}$ act on a different session than u, or (*iii*) do internal actions.

We now introduce the final ingredient for honesty, that is the notion of *readiness* of a participant.

---

[1] The side contition "$u \notin \vec{v}$" of Definition 4.4 deals with cases like $S_0 = (s)\,(\mathtt{A}[\mathsf{do}^s_{\mathtt{B}}\, \mathsf{int}])$ and $S = S_0 \mid s[\mathtt{A}\langle \mathsf{B!int}\rangle \mid \ldots] \mid \ldots$: without the side condition, $\mathrm{PRS}^u_{\mathtt{A}}(S_0) = \{\{(\mathtt{B}, \mathsf{int})\}\}$ — hence, by Def. 4.6, $\mathtt{A}$ would result to be ready in S.

Definition 4.6 (Readiness)**.** *We say that* $\mathtt{A}$ *is ready in* S *iff, whenever* $S \equiv (\vec{u},\vec{\mathtt{b}})S_0$ *for some* $\vec{u},\vec{\mathtt{b}}$ *and* $S_0 \equiv s[\mathtt{A}\langle c\rangle \mid \cdots] \mid \cdots$*, the following holds:*

$$\exists \mathscr{X} \in \mathrm{CRS}(c):\ \big((\mathtt{B},\mathtt{e}) \in \mathscr{X} \implies (\mathtt{B},\mathtt{e}) \in \mathrm{WPRS}^{s}_{\mathtt{A}}(S_1)\big)$$

Definition 4.6 says that a participant $\mathtt{A}$ is *ready* in a system S whenever its process ready sets for a session s will eventually contain all the participant/sort pairs of one of the contract ready sets of $\mathtt{A}$'s contract in s. When a participant $\mathtt{A}$ is "ready", then, for any of its contracts c, the CO_2 process of $\mathtt{A}$ is (eventually) able to fulfil at least the interactions in c's prefix.

Example 4.7. *We have seen that, after fusion of the latent contracts of* S_1 *(in Ex. 3.3) we obtain:*

$$(s)S_1' \quad\equiv\quad (s)\big(\mathtt{A}[P_{\mathtt{A}}\sigma\pi] \;\mid\; s[T_{\mathtt{AB_1B_2}} \mid \mathtt{Q}(T_{\mathtt{AB_1B_2}})] \;\mid\; \mathtt{B}_1[P_{\mathtt{B}_1}\sigma\pi] \;\mid\; \mathtt{B}_2[P_{\mathtt{B}_2}\sigma\pi]\big)$$

where the substitutions σ *and* π *are also from Ex. 3.3. Let us define the processes (after substitutions):*

$$\begin{array}{rcl}
P_{\mathtt{A}}\sigma\pi & = & \mathsf{do}^{s}_{\mathtt{B}_1}\,\mathsf{req}\,.\,\mathsf{do}^{s}_{\mathtt{B}_2}\,\mathsf{req}\,.\,\mathsf{do}^{s}_{\mathtt{B}_1}\,\mathsf{quote}\,.\,\big(\mathsf{do}^{s}_{\mathtt{B}_1}\,\mathsf{order}\,.\,\mathsf{do}^{s}_{\mathtt{B}_2}\,\mathsf{ok} \;+\; \mathsf{do}^{s}_{\mathtt{B}_1}\,\mathsf{bye}\,.\,\mathsf{do}^{s}_{\mathtt{B}_2}\,\mathsf{bye}\big)\\
P_{\mathtt{B}_1}\sigma\pi & = & \tau\,.\,\mathsf{do}^{s}_{\mathtt{A}}\,\mathsf{req}\,.\,\mathsf{do}^{s}_{\mathtt{A}}\,\mathsf{quote}\,.\,\mathsf{do}^{s}_{\mathtt{A}}\,\mathsf{order}\\
P_{\mathtt{B}_2}\sigma\pi & = & \mathsf{do}^{s}_{\mathtt{A}}\,\mathsf{req}\,.\,\big(\mathsf{do}^{s}_{\mathtt{B}_1}\,\mathsf{ok}\,.\,\mathsf{do}^{s}_{\mathtt{A}}\,\mathsf{ok} \;+\; \mathsf{do}^{s}_{\mathtt{B}_1}\,\mathsf{bye}\,.\,\mathsf{do}^{s}_{\mathtt{A}}\,\mathsf{bye}\big)
\end{array}$$

Thus, we have:

$$\begin{array}{rcccl}
\mathrm{PRS}^{s}_{\mathtt{A}}(S_1') & = & \{(\mathtt{B}_1,\mathsf{req})\} & = & \mathrm{WPRS}^{s}_{\mathtt{A}}(S_1')\\
\mathrm{PRS}^{s}_{\mathtt{B}_1}(S_1') & = & \varnothing \neq \{(\mathtt{A},\mathsf{req})\} & = & \mathrm{WPRS}^{s}_{\mathtt{B}_1}(S_1')\\
\mathrm{PRS}^{s}_{\mathtt{B}_2}(S_1') & = & \{(\mathtt{A},\mathsf{req})\} & = & \mathrm{WPRS}^{s}_{\mathtt{B}_2}(S_1')
\end{array}$$

Note that the τ *prefix in* $P_{\mathtt{B}_1}$ *prevents* $\mathtt{B}_1$ *from interacting immediately with* $\mathtt{A}$ *on session* s*, although it is "weakly ready" to do it. Hence, considering that the weak process ready sets of each participant in* S_1' *match their respective contract ready sets in* $T_{\mathtt{AB_1B_2}}$ *(Example 4.3) according to Definition 4.6 we have that participants* $\mathtt{A}$*,* $\mathtt{B}_1$ *and* $\mathtt{B}_2$ *are all ready in* $(s)S_1'$*.*

Before defining honesty formally, we need to characterise the class of systems for which this concept is meaningful, i.e., those systems where a participant is not (yet) involved in latent contracts nor active sessions.

Definition 4.8 (Initial System)**.** *A* CO_2 *system* S *is* $\mathtt{A}$-initial *if* S *has no sub-term of the form* $\downarrow_{\_}\ \mathtt{A}$ *says* c *or* $\mathtt{A}\langle c\rangle$ *with* $c \not\equiv \mathbf{0}$*. A* CO_2 *system* S *is* initial *when it is* $\mathtt{A}$*-initial for each participant* $\mathtt{A}$ *in* S*.*

Definition 4.9 (Honesty)**.** *We say that* $\mathtt{A}[P]$ *is* honest *iff, for all* $\mathtt{A}$*-initial* $S \equiv (\vec{u},\vec{\mathtt{b}})\,(\mathtt{A}[P] \mid S_0)$ *s.t.* $S \rightarrow^{*} S'$*,* $\mathtt{A}$ *is ready in* S'*.*

A process $\mathtt{A}[P]$ is said to be honest when, for all contexts and reductions that $\mathtt{A}[P]$ may be engaged in, $\mathtt{A}$ is persistently ready. In other words, there is a continuous correspondence between the interactions exposed in the contract ready sets and the process ready sets of the possible reductions of any system involving $\mathtt{A}[P]$. The definition rules out contexts with latent/stipulated contracts of $\mathtt{A}$, otherwise $\mathtt{A}$ could be made trivially dishonest, e.g. by inserting a latent contract $\downarrow_u \mathtt{A}$ *says* c that $\mathtt{A}$ cannot fulfil.

Example 4.10. *Consider the process* $\mathtt{B}_1[\mathsf{tell}_{\mathtt{A}} \downarrow_y c_{\mathtt{B}_1}\,.\,P_{\mathtt{B}_1}]$ *of system* S_1*, as defined in Examples 3.3 and 4.7. We show that this process is* not *honest. In fact,* S_1 *can reduce as* $S_1 \rightarrow^{*} (s)S_1' \rightarrow^{*} (s)S_1''$*, where:*

$$\begin{array}{rl}
(s)S_1'' = (s)\Big(& \mathtt{A}\big[\mathsf{do}^{s}_{\mathtt{B}_1}\,\mathsf{order}\,.\,\mathsf{do}^{s}_{\mathtt{B}_2}\,\mathsf{ok} \;+\; \mathsf{do}^{s}_{\mathtt{B}_1}\,\mathsf{bye}\,.\,\mathsf{do}^{s}_{\mathtt{B}_2}\,\mathsf{bye}\big]\\
& \mid\; s\big[\;\mathtt{A}\langle \mathtt{B}_1?\mathsf{order};\mathtt{B}_2!\mathsf{ok} + \mathtt{B}_1?\mathsf{bye};\mathtt{B}_2!\mathsf{bye}\rangle\\
& \qquad \mid\; \mathtt{B}_1\langle \mathtt{B}_2!\mathsf{ok};\mathtt{A}!\mathsf{order} \oplus \mathtt{B}_2!\mathsf{bye};\mathtt{A}!\mathsf{bye}\rangle \;\mid\; \mathtt{B}_2\langle \mathtt{B}_1?\mathsf{ok};\mathtt{A}?\mathsf{ok} + \mathtt{B}_1?\mathsf{bye};\mathtt{A}?\mathsf{bye}\rangle\\
& \qquad \mid (\mathtt{AB}_1):[\,] \mid (\mathtt{B}_1\mathtt{A}):[\,] \mid (\mathtt{AB}_2):[\,] \mid (\mathtt{B}_2\mathtt{A}):[\,] \mid (\mathtt{B}_1\mathtt{B}_2):[\,] \mid (\mathtt{B}_2\mathtt{B}_1):[\,]\;\big]\\
& \mid\; \mathtt{B}_1[\mathsf{do}^{s}_{\mathtt{A}}\,\mathsf{order}] \;\mid\; \mathtt{B}_2\big[\mathsf{do}^{s}_{\mathtt{B}_1}\,\mathsf{ok}\,.\,\mathsf{do}^{s}_{\mathtt{A}}\,\mathsf{ok} \;+\; \mathsf{do}^{s}_{\mathtt{B}_1}\,\mathsf{bye}\,.\,\mathsf{do}^{s}_{\mathtt{A}}\,\mathsf{bye}\big]\Big)
\end{array}$$

At this point, we see that there is a problem in the implementation of $\mathtt{B}_1$*: it does not notify the other buyer before making an order. In fact,* $\mathtt{B}_1$*'s process is trying to perform* $\mathsf{do}^s_{\mathtt{A}}\,\mathsf{order}$*, but its contract requires that* $\mathsf{do}^s_{\mathtt{B}_2}\,\mathsf{ok}$ *is performed first (or* $\mathsf{do}^s_{\mathtt{B}_2}\,\mathsf{bye}$*, if the quote is rejected). This is reflected by the mismatch between* $\mathtt{B}_1$*'s process ready set in* S_1''*, and its contract ready sets, in session s:*

$$\begin{array}{rcl} \mathrm{PRS}^s_{\mathtt{B}_1}(S_1'') & = & \{\{(\mathtt{A},\mathsf{order})\}\} \\ \mathrm{CRS}(\mathtt{B}_2!\mathsf{ok};\mathtt{A}!\mathsf{order}\oplus\mathtt{B}_2!\mathsf{bye};\mathtt{A}!\mathsf{bye}) & = & \{\{(\mathtt{B}_2,\mathsf{ok})\},\{(\mathtt{B}_2,\mathsf{bye})\}\} \end{array}$$

In terms of the above definitions, there exists a system S_1 *— containing* $\mathtt{B}_1[\mathsf{tell}_{\mathtt{A}}\downarrow_y c_{\mathtt{B}_1}\,.\,P_{\mathtt{B}_1}]$ *— that reduces to a* $(s)S_1''$ *where* $\mathtt{B}_1$ *is not ready (Definition 4.6). Therefore,* $\mathtt{B}_1[\mathsf{tell}_{\mathtt{A}}\downarrow_y c_{\mathtt{B}_1}\,.\,P_{\mathtt{B}_1}]$ *is not honest. In fact,* $\mathtt{B}_1$ *is culpable in* $(s)S_1''$*, according to Definition 4.1.*

As in [1], the definition of honesty subsumes a *fair* scheduler, eventually allowing participants to fire persistently (weakly) enabled do actions.

Honesty is not decidable in general [4], but for a bilateral contract model it has been approximated either via an abstract semantics [4] or a type discipline [1] for CO_2. We believe that these approximations may be easily adapted to our setting (see Section 6 for more details).

5 Results

We now give the main properties of our framework. We ensure that two basic features of CO_2 hold in our multiparty adaptation: the state of a session always allows to establish who is responsible for making the system progress (Th. 5.1) and honest participants can always exculpate themselves (Th. 5.3). We then formalise a link between the honesty of participants, and two key properties borrowed from the session types setting: Th. 5.4 introduces session fidelity in CO_2; and Th. 5.5 introduces a notion of progress in CO_2, based on the progress of the contractual agreement (and its choreography).

Theorem 5.1 (Unambiguous Culpability)**.** *Given an initial* CO_2 *system S, if* $S \to^* S' \equiv (\vec{u},\vec{b})\,(s[T] \mid \cdots)$ *such that* $T \not\equiv \mathbf{0}$*, then there exists at least one culpable participant in* S'*.*

Theorem 5.1 says that in an active session established through a fuse reduction, there is always at least one participant $\mathtt{A}[P]$ who leads the next interaction. Thus, if a corresponding $\mathsf{do}^s_{\mathtt{B}}\mathsf{e}$ prefix is not in P, S may get stuck, and $\mathtt{A}$ is culpable.

Example 5.2. *Consider the system* S_1'' *in Example 4.10, and the system of contracts in its session s:*

$$\begin{array}{rcl} T_s & = & \mathtt{A}\langle \mathtt{B}_1?\mathsf{order};\mathtt{B}_2!\mathsf{ok}+\mathtt{B}_1?\mathsf{bye};\mathtt{B}_2!\mathsf{bye}\rangle \\ & & \mid\ \mathtt{B}_1\langle \mathtt{B}_2!\mathsf{ok};\mathtt{A}!\mathsf{order}\oplus\mathtt{B}_2!\mathsf{bye};\mathtt{A}!\mathsf{bye}\rangle\ \mid\ \mathtt{B}_2\langle \mathtt{B}_1?\mathsf{ok};\mathtt{A}?\mathsf{ok}+\mathtt{B}_1?\mathsf{bye};\mathtt{A}?\mathsf{bye}\rangle \\ & & \mid\ (\mathtt{AB}_1):[\,]\mid(\mathtt{B}_1\mathtt{A}):[\,]\mid(\mathtt{AB}_2):[\,]\mid(\mathtt{B}_2\mathtt{A}):[\,]\mid(\mathtt{B}_1\mathtt{B}_2):[\,]\mid(\mathtt{B}_2\mathtt{B}_1):[\,] \end{array}$$

We have $T_s \xrightarrow{\mathtt{B}_1\leftrightharpoons\mathtt{B}_2:\mathsf{ok}}$ *and* $T_s \xrightarrow{\mathtt{B}_1\leftrightharpoons\mathtt{B}_2:\mathsf{bye}}$*. Hence,* $\mathtt{B}_1$ *is responsible for the next interaction, and culpable for* S_1'' *being stuck.*

Theorem 5.3 (Exculpation)**.** *Given an* $\mathtt{A}$*-initial* CO_2 *system* S_0 *with* $\mathtt{A}[P]$ *honest, whenever* $S_0 \to^* S \equiv (\vec{u},\vec{\mathtt{a}})\,(s[T] \mid S_1)$ *and* $\mathtt{A}$ *is culpable in S at session s, there exist* $\mathtt{B}$ *and* e *such that:* $S \xrightarrow{\mathtt{A}:\,p}{}^{*} \xrightarrow{\mathtt{A}:\,\mathsf{do}^{\mathtt{B}}_s\,\mathsf{e}}$ *where* $p = \tau$ *or* $p = \mathsf{tell}_{\_}\downarrow_{\_}\,\_$*.*

Theorem 5.3 follows from the definition of honesty, formalising that honest participants can always overcome their culpability, by firing their contractual do actions (possibly after advertising other contracts or performing some internal actions).

Theorem 5.4 (Fidelity)**.** *For all initial systems S with only honest participants, if S is such that $S \rightarrow^* S' \equiv (\vec{u},\vec{\mathtt{a}})\,(\mathtt{A}[P] \mid s[T] \mid S_0)$, then $(S' \xrightarrow{\neq(\mathtt{A}:\,\mathsf{do}^{s})}{}^{*} \xrightarrow{\mathtt{A}:\,\mathsf{do}^{s}_{\mathtt{B}}\,\mathsf{e}}) \iff (T \xrightarrow{\mathtt{A}\Leftarrow\mathtt{B}:\mathtt{e}})$ (where $\xrightarrow{\neq(\mathtt{A}:\,\mathsf{do}^{u})}{}^{*}$, as in Def. 4.5, intuitively stands for any reduction* not *involving session s).*

Theorem 5.4 says that each (honest) participant will strictly adhere to its contracts, once they have been fused in a session. It follows directly from the semantics of CO_2 (that forbid non-contractual do prefixes to be fired) and from the definition of honesty.

Theorem 5.5 below introduces the notion of global progress, which is slightly different from the contractual progress. In fact, progress in CO_2 is only meaningful *after* a session has been established, and thus a culpable participant exists. A system without sessions may not progress because a set of compliant contracts cannot be found, or a fuse prefix is not enabled. In both cases, no participant may be deemed culpable, and thus responsible for the next move. However, the system may progress again if other (honest) participants join it, allowing a session to be established.

Theorem 5.5 (Global Progress)**.** *Given an initial CO_2 system S_0 with only honest participants, if $S_0 \rightarrow^* S \equiv (\vec{u},\vec{\mathtt{a}})\,(s[T] \mid S_1)$ with $T \not\equiv \mathbf{0}$, then $S \rightarrow$.*

Theorem 5.5 follows from the definition of honesty (i.e., participants are always ready to fulfil their contracts), the fact that contract compliance guarantees contractual progress [15], Theorem 5.3, and the semantics of CO_2 (in particular, rule [CO$_2$-DO]). This result also holds for systems where a process takes part in multiple sessions: the honesty of all participants guarantees that all sessions will be completed.

Example 5.6. *We now give a simple example on a system with multiple sessions. We show how a seemingly honest process ($\mathtt{B}$) could be deemed culpable due to the unexpected behaviour of other participants, and how honest participants guarantee progress of the whole system. Consider:*

$$
\begin{aligned}
S = (x,y,z,w)\big(\; &\mathtt{A}[\mathsf{tell}_{\mathtt{A}} \downarrow_x (\mathtt{B}!\mathsf{int})\,.\,\mathsf{fuse}\,.\,\mathsf{fuse}] \;\;\mid\;\; \mathtt{B}\big[\mathsf{tell}_{\mathtt{A}} \downarrow_y (\mathtt{A}?\mathsf{int})\,.\,\mathsf{tell}_{\mathtt{A}} \downarrow_z (\mathtt{C}!\mathsf{bool})\,.\,\mathsf{do}^{y}_{\mathtt{A}}\,\mathsf{int}\,.\,\mathsf{do}^{z}_{\mathtt{C}}\,\mathsf{bool}\big] \\
&\mid\;\; \mathtt{C}[\mathsf{tell}_{\mathtt{A}} \downarrow_w (\mathtt{B}?\mathsf{bool})\,.\,\mathsf{do}^{w}_{\mathtt{B}}\,\mathsf{bool}]\;\big)
\end{aligned}
$$

After all four contracts have been advertised to $\mathtt{A}$ and fuse*d, the system reduces to:*

$$
\begin{aligned}
S' = (s_1,s_2)\big(\; &\mathtt{A}[\mathbf{0}] \;\;\mid\;\; \mathtt{B}[\mathsf{do}^{s_1}_{\mathtt{A}}\,\mathsf{int}\,.\,\mathsf{do}^{s_2}_{\mathtt{C}}\,\mathsf{bool}] \;\;\mid\;\; \mathtt{C}[\mathsf{do}^{s_2}_{\mathtt{B}}\,\mathsf{bool}] \\
&\mid\;\; s_1[\mathtt{A}\langle\mathtt{B}?\mathsf{int}\rangle \mid \mathtt{B}\langle\mathtt{A}!\mathsf{int}\rangle \mid (\mathtt{AB}):[] \mid (\mathtt{BA}):[]] \;\;\mid\;\; s_2[\mathtt{B}\langle\mathtt{C}!\mathsf{bool}\rangle \mid \mathtt{C}\langle\mathtt{B}?\mathsf{bool}\rangle \mid (\mathtt{BC}):[] \mid (\mathtt{CB}):[]]\;\big)
\end{aligned}
$$

Even if both sessions s_1 and s_2 enjoy contractual progress, S' is stuck: $\mathtt{A}$ does not perform the promised action, thus remaining culpable in s_1; $\mathtt{B}$ is stuck waiting in s_1, thus remaining culpable in s_2.[2] *Indeed, neither $\mathtt{A}$ nor $\mathtt{B}$ are ready in S', and thus they are not honest in S. Hence, global progress is not guaranteed. Let us now consider the following variant of S, where all participants are honest:*

$$
\begin{aligned}
\hat{S} = (x,y,z,w)\big(\; &\mathtt{A}[(\mathsf{tell}_{\mathtt{A}} \downarrow_x (\mathtt{B}!\mathsf{int})\,.\,\mathsf{do}^{x}_{\mathtt{B}}\,\mathsf{int}) \mid \mathsf{fuse} \mid \mathsf{fuse}] \;\;\mid\;\; \mathtt{C}[\mathsf{tell}_{\mathtt{A}} \downarrow_w (\mathtt{B}?\mathsf{bool})\,.\,\mathsf{do}^{w}_{\mathtt{B}'}\,\mathsf{bool}] \\
&\mid\;\; \mathtt{B}\big[\mathsf{tell}_{\mathtt{A}} \downarrow_y (\mathtt{A}?\mathsf{int})\,.\,\mathsf{tell}_{\mathtt{A}} \downarrow_z (\mathtt{C}!\mathsf{bool})\,.\,\big(\mathsf{do}^{y}_{\mathtt{A}}\,\mathsf{int}\,.\,\mathsf{do}^{z}_{\mathtt{C}}\,\mathsf{bool} + \tau\,.\,\big(\mathsf{do}^{y}_{\mathtt{A}}\,\mathsf{int} \mid \mathsf{do}^{z}_{\mathtt{C}}\,\mathsf{bool}\big)\big)\big]\;\big)
\end{aligned}
$$

In this case, $\mathtt{A}$ will respect its contractual duties, while $\mathtt{B}$ will be ready to fulfil its contracts on both sessions — even if one is not activated, or remains stuck (here, τ represents an internal action, e.g., a timeout: if the first $\mathsf{do}^{y}_{\mathtt{A}}\,\mathsf{int}$ cannot reduce, $\mathtt{B}$ falls back to running the sessions in parallel). The honesty of all participants in $\hat{S}$ guarantees that, once a session is active, it will reach its completion.

---

[2] In this case, $\mathtt{B}$ is deemed culpable in s_2 because its implementation did not expect $\mathtt{A}$ to misbehave.

6 Concluding remarks

In this work, we investigated the combination of the contract-oriented calculus CO_2 with a contract model that fulfils two basic design requirements: (i) it supports multiparty agreements, and (ii) it provides an explicit description of the choreography that embodies each agreement. These requirements prompted us towards the well-established results from the session types setting — in particular, regarding the interplay between a well-formed global type and its corresponding local behaviours. We introduced the concepts of global progress and session fidelity in CO_2, also inspired from the analogous concepts in theories based on session types. We built our framework upon a simple version of session types, and yet it turns out to be quite flexible, allowing for sessions where the number of participants is not known beforehand.

Related work. The origin of CO_2 goes back to [5]. The calculus was generalised in [2, 3] to suit different contract models (e.g., contracts as processes or logic formulæ). In [4], it has been instantiated to a theory of bilateral contracts inspired by [9]. A negative result in [4] is that the problem of honesty (Section 4) is not decidable. A type system for CO_2 processes providing a decidable approximation of honesty was introduced in [1]. This result relies on the product between a finite state system (approximating contracts) and a Basic Parallel Process (approximating a CO_2 process). Considering that the systems of contracts in this work form a (strict) subset of the local/global types in [10], for which each configuration is reachable by a 1-buffer execution, we believe that the type system in [1] may be adapted to our setting.

The seminal top-down approach of multiparty session types has been first described in [11]. In summary, the framework works as follows: designers specify a choreography (i.e., a global type), which is then projected onto local behaviours (i.e., local types), which in turn are used to type-check processes. A dual approach was introduced in [15]: from a set of local types it is possible to synthesise a choreography (i.e., a global type). This is precisely the result we use as a basis for our definition of compliant contracts.

The semantic correspondence between global types and projected local behaviours has been investigated in [12, 8]. To the best of our knowledge, no other contract model besides ours is based on explicit choreography synthesis. A related approach is presented in [7]: multiple contracts are considered compliant when their composition (i.e., the system of contracts) guarantees completion. In our work, progress (subsuming completion) is provided by the synthesis of a global type. In [6], contracts are considered compliant when their composition adheres to a predetermined choreography; in our framework, however, no choreography is assumed beforehand.

Future work. We plan to extend our work so to offer even more flexibility. For example, by introducing a parameterised fuse primitive which starts a session according to different criteria, when more than one agreement is possible (as in our introductory example). For instance, one could choose the agreement involving the most (or least) number of participants. These criteria may be based on a semantic characterisation of global types, e.g., as the ones in [12, 8]. We also plan to study the possibility for a participant to be involved in a session under multiple contracts, e.g., a bank advertising two services, and a customer publishing a contract which uses both of them in a well-formed choreography.

Another research direction is the concept of "group honesty". In fact, the current definition of honesty is quite strict: it basically verifies each participant in isolation, thus providing a sufficient (but not necessary) condition for progress. Consider, for example, a CO_2 system like:

$$S \;=\; (x,y)\Big(\mathtt{A}\big[\mathsf{tell}_{\mathtt{A}} \downarrow_x (\mathtt{B}!\mathsf{int} \oplus \mathtt{B}!\mathsf{bool})\,.\,\mathsf{fuse}\,.\,\mathsf{do}^{x}_{\mathtt{B}}\,\mathsf{int}\big] \;\mid\; \mathtt{B}\big[\mathsf{tell}_{\mathtt{A}} \downarrow_y (\mathtt{A}?\mathsf{int} + \mathtt{A}?\mathsf{bool})\,.\,\mathsf{do}^{y}_{\mathtt{A}}\,\mathsf{int}\big]\Big)$$

`B` is dishonest, since it is not ready for the bool branch of its contract. However, the system S has progress: when `B` establishes a session with `A`, the latter will never take the bool branch; hence, `B` will not remain culpable. This kind of "group honesty" may be used to validate (sub-)systems of participants developed by the same organization: it would ensure that they never "cheat each other", and are collectively honest when deployed in any context. Furthermore, the group honesty of all participants in a system S may turn out to be a necessary condition for the global progress of S.

Acknowledgements. We would like to thank Massimo Bartoletti, Emilio Tuosto, and Roberto Zunino for their valuable advice, discussions and comments. We would also like to thank the anonymous reviewers for their suggestions. This work is partially supported by Aut. Region of Sardinia under grants L.R.7/2007 CRP-17285 (TRICS) and P.I.A. 2010 project "Social Glue", and by MIUR PRIN 2010-11 project "Security Horizons", and by EU COST Action IC1201 (BETTY).

References

[1] M. Bartoletti, A. Scalas, E. Tuosto & R. Zunino (2013): *Honesty by Typing*. In: *FMOODS/FORTE*, pp. 305–320. Available at `http://dx.doi.org/10.1007/978-3-642-38592-6_21`.

[2] M. Bartoletti, E. Tuosto & R. Zunino (2011): *Contracts in distributed systems*. In: *ICE*, pp. 130–147. Available at `http://dx.doi.org/10.4204/EPTCS.59.11`.

[3] M. Bartoletti, E. Tuosto & R. Zunino (2012): *Contract-oriented Computing in CO_2*. *Scientific Annals in Comp. Sci.* 22(1), pp. 5–60. Available at `http://dx.doi.org/10.7561/SACS.2012.1.5`.

[4] M. Bartoletti, E. Tuosto & R. Zunino (2012): *On the Realizability of Contracts in Dishonest Systems*. In: *COORDINATION*. Available at `http://dx.doi.org/10.1007/978-3-642-30829-1_17`.

[5] M. Bartoletti & R. Zunino (2010): *A Calculus of Contracting Processes*. In: *LICS*. Available at `http://doi.ieeecomputersociety.org/10.1109/LICS.2010.25`.

[6] M. Bravetti, I. Lanese & G. Zavattaro (2009): *Contract-Driven Implementation of Choreographies*. In: *TGC*, *LNCS* 5474, doi:10.1007/978-3-642-00945-7_1. Available at `http://dx.doi.org/10.1007/978-3-642-00945-7_1`.

[7] M. Bravetti & G. Zavattaro (2008): *A Foundational Theory of Contracts for Multi-party Service Composition*. *Fundam. Inform.* 89(4). Available at `http://iospress.metapress.com/content/x7613955410077k2/`.

[8] G. Castagna, M. Dezani-Ciancaglini & L. Padovani (2012): *On Global Types and Multi-Party Session*. *Logical Methods in Comp. Sci.* 8(1). Available at `http://dx.doi.org/10.2168/LMCS-8(1:24)2012`.

[9] G. Castagna, N. Gesbert & L. Padovani (2009): *A theory of contracts for Web services*. *ACM Trans. on Prog. Lang. and Sys.* 31(5), doi:10.1145/1538917.1538920. Available at `http://doi.acm.org/10.1145/1538917.1538920`.

[10] P.M. Deniélou & N. Yoshida (2012): *Multiparty Session Types Meet Communicating Automata*. In: *ESOP*. Available at `http://dx.doi.org/10.1007/978-3-642-28869-2_10`.

[11] K. Honda, N. Yoshida & M. Carbone (2008): *Multiparty asynchronous session types*. In: *POPL*. Available at `http://doi.acm.org/10.1145/1328438.1328472`.

[12] I. Lanese, C. Guidi, F. Montesi & G. Zavattaro (2008): *Bridging the Gap between Interaction- and Process-Oriented Choreographies*. SEFM '08, doi:10.1109/SEFM.2008.11. Available at `http://dx.doi.org/10.1109/SEFM.2008.11`.

[13] J. Lange & A. Scalas (2013): *Choreography Synthesis as Contract Agreement*. In: *ICE*. To appear.

[14] J. Lange & A. Scalas (2013): *Choreography Synthesis as Contract Agreement + Appendix*. Available at `http://tcs.unica.it/publications`.

[15] J. Lange & E. Tuosto (2012): *Synthesising Choreographies from Local Session Types*. In: *CONCUR*. Available at `http://dx.doi.org/10.1007/978-3-642-32940-1_17`.

[16] OASIS (2012): *Reference Architecture Foundation for Service Oriented Architecture*. Comm. Spec. 01, v.1.0. Available at `http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/cs01/soa-ra-v1.0-cs01.html`.

A Semantics of Local Types

Below are the rest of the semantic rules for local types / contracts (extending the ones on [13]). λ ranges over the labels $\mathtt{A} \to \mathtt{B} : \mathtt{e}$ and $\mathtt{A} \leftarrow \mathtt{B} : \mathtt{e}$, where the former means that a message of sort $\mathtt{e}$ is sent by participant $\mathtt{A}$ to $\mathtt{B}$, and the latter means that a message of sort $\mathtt{e}$ is received by $\mathtt{A}$ from $\mathtt{B}$.

$$\mu\,\mathbf{x}.c \equiv c\,\{^{\mathbf{x}}/_{\mu\,\mathbf{x}.c}\} \qquad\qquad \text{commutative monoidal laws for } \mid \text{ and } \mathbf{0}$$

$$T_1 \xrightarrow{\lambda} T_1' \quad\Rightarrow\quad T_1 \mid T_2 \xrightarrow{\lambda} T_1' \mid T_2 \qquad\qquad T_2 \equiv T_1 \xrightarrow{\lambda} T_1' \equiv T_2' \quad\Rightarrow\quad T_2 \xrightarrow{\lambda} T_2'$$

We define $\mathbf{0} = \bigoplus_{i\in\varnothing} \underline{\mathtt{a}}_{\mathtt{i}}!\mathsf{e}_{\mathsf{i}};c_i = \sum_{i\in\varnothing} \underline{\mathtt{a}}?\mathsf{e}_{\mathsf{i}};c_i$

B Synthesis of global types

$$[;]\ \frac{\Gamma \vdash \mathtt{A}\langle c\rangle \mid \mathtt{B}\langle c'\rangle \mid T \blacktriangleright \mathcal{G} \qquad T \not\updownarrow}{\Gamma \vdash \mathtt{A}\langle \mathtt{B}!\mathsf{e};c\rangle \mid \mathtt{B}\langle \mathtt{A}?\mathsf{e};c'\rangle \mid T \blacktriangleright \mathtt{A}\to\mathtt{B}:\mathtt{e};\mathcal{G}} \qquad [\,|\,]\ \frac{\circ \vdash T \blacktriangleright \mathcal{G} \qquad \circ \vdash T' \blacktriangleright \mathcal{G}'}{\Gamma \vdash T \mid T' \blacktriangleright \mathcal{G} \mid \mathcal{G}'}$$

$$[\oplus]\ \frac{\Gamma \vdash \mathtt{A}\langle c\rangle \mid T \blacktriangleright \mathcal{G} \qquad \Gamma \vdash \mathtt{A}\langle c'\rangle \mid T \blacktriangleright \mathcal{G}' \qquad T \not\updownarrow}{\Gamma \vdash \mathtt{A}\langle c \oplus c'\rangle \mid T \blacktriangleright \mathcal{G} + \mathcal{G}'} \qquad [+]\ \frac{\Gamma \vdash \mathtt{B}\langle c\rangle \mid T \blacktriangleright \mathcal{G}}{\Gamma \vdash \mathtt{B}\langle c + c'\rangle \mid T \blacktriangleright \mathcal{G}}$$

$$[\mu]\ \frac{\exists 1 \le i,j \le k\,.\,(\mathtt{A_i}\langle c_i\rangle \mid \mathtt{A_j}\langle c_j\rangle)\updownarrow \qquad \Gamma\cdot(\mathtt{A}_1,\mathbf{x}_1):\mathbf{x},\ldots,(\mathtt{A}_k,\mathbf{x}_k):\mathbf{x} \vdash \mathtt{A_1}\langle c_1\rangle \mid \ldots \mid \mathtt{A_k}\langle c_k\rangle \blacktriangleright \mathcal{G}}{\Gamma \vdash \mathtt{A_1}\langle \mu\,\mathbf{x}_1.c_1\rangle \mid \ldots \mid \mathtt{A_k}\langle \mu\,\mathbf{x}_k.c_k\rangle \blacktriangleright \mu\,\mathbf{x}.\mathcal{G}}$$

$$[\mathbf{x}]\ \frac{\forall 1 \le i \le k\,.\,\Gamma(\mathtt{A_i},\mathbf{x}_i) = \mathbf{x}}{\Gamma \vdash \mathtt{A_1}\langle \mathbf{x}_1\rangle \mid \ldots \mid \mathtt{A_k}\langle \mathbf{x}_k\rangle \blacktriangleright \mathbf{x}} \qquad [eq]\ \frac{T' \equiv T' \qquad \Gamma \vdash T' \blacktriangleright \mathcal{G}}{\Gamma \vdash T \blacktriangleright \mathcal{G}} \qquad [\mathbf{0}]\ \frac{\forall \mathtt{n} \in \mathscr{P}(T)\,.\,T(\mathtt{n}) = \mathbf{0}}{\Gamma \vdash T \blacktriangleright \mathbf{0}}$$

We define the *ready set* of a system as follows:

$$\mathtt{R}(T) = \begin{cases} \{\mathtt{A}\leftarrow\mathtt{B_i} \mid i \in I\} \cup \mathtt{R}(T') & \text{if } T \equiv \mathtt{A}\langle \sum_{i\in I} \mathtt{B}?\mathsf{e}_{\mathsf{i}};c_i\rangle \mid T' \\ \{\mathtt{A}\to\mathtt{B_i} \mid i \in I\} \cup \mathtt{R}(T') & \text{if } T \equiv \mathtt{A}\langle \bigoplus_{i\in I} \mathtt{B_i}!\mathsf{e}_{\mathsf{i}};c_i\rangle \mid T' \\ \{\mathtt{A}\to\mathtt{B}\} \cup \mathtt{R}(T') & \text{if } T \equiv (\mathtt{AB}):\mathtt{e}\cdot\rho \mid T' \\ \varnothing & \text{if } T \equiv \mathbf{0} \end{cases}$$

We overload $\mathtt{R}(\_)$ on behaviours as expected, and define $T\updownarrow \iff \exists \mathtt{A}\to\mathtt{B} : \mathtt{A}\to\mathtt{B} \in \mathtt{R}(T) \wedge \mathtt{B}\leftarrow\mathtt{A} \in \mathtt{R}(T)$; we write $T\not\updownarrow$ if $T\updownarrow$ does not hold.

C Semantics of CO$_2$

Below is the rest of the semantic rules for CO$_2$ (extending the ones on page 6).

$$[\text{CO}_2\text{-Tau}] \quad \mathtt{A}[\tau\,.\,P + P' \mid Q] \;\to\; \mathtt{A}[P \mid Q]$$

$$[\text{CO}_2\text{-Def}]\ \frac{X(\vec{u},\vec{\mathtt{a}}) := P \qquad (\vec{z},\vec{\mathtt{c}})\,\big(\mathtt{A}\big[P\,\{^{\vec{v}}/_{\vec{u}}\}\,\{^{\vec{\underline{\mathtt{b}}}}/_{\vec{\mathtt{a}}}\} \mid Q\big] \mid S\big) \;\to\; S'}{(\vec{z},\vec{\mathtt{c}})\,\big(\mathtt{A}\big[X(\vec{v},\vec{\underline{\mathtt{b}}}) \mid Q\big] \mid S\big) \;\to\; S'}$$

$$[\text{CO}_2\text{-Par}]\ \frac{S \to S'}{S \mid S'' \to S' \mid S''} \qquad\qquad [\text{CO}_2\text{-Del}]\ \frac{S \to S'}{(\vec{u},\vec{\mathtt{a}})S \to (\vec{u},\vec{\mathtt{a}})S'}$$

Structural congruence for CO$_2$ (Z, Z', Z'' range over processes, systems, or latent contracts):

$$(\vec{u},\vec{\mathtt{b}})\mathtt{A}[(\vec{v},\vec{\mathtt{c}})P] \equiv (\vec{u},\vec{\mathtt{b}})(\vec{v},\vec{\mathtt{c}})\mathtt{A}[P] \qquad \mathtt{A}[\mathbf{0}] \equiv \mathbf{0} \qquad \mathtt{A}[K] \mid \mathtt{A}[K'] \equiv \mathtt{A}[K \mid K']$$
$$Z \mid \mathbf{0} \equiv Z \qquad Z \mid Z' \equiv Z' \mid Z \qquad (Z \mid Z') \mid Z'' \equiv Z \mid (Z' \mid Z'')$$
$$Z \mid (\vec{u},\vec{\mathtt{a}})Z' \equiv (\vec{u},\vec{\mathtt{a}})(Z \mid Z') \text{ if } \vec{u} \cap \mathrm{fnv}(Z) = \vec{\mathtt{a}} \cap \mathrm{fnv}(Z) = \varnothing$$
$$(\vec{u},\vec{\mathtt{a}})(\vec{v},\vec{\mathtt{b}})Z \equiv (\vec{v},\vec{\mathtt{b}})(\vec{u},\vec{\mathtt{a}})Z \qquad (\vec{u},\vec{\mathtt{a}})(\vec{v},\vec{\mathtt{b}})Z \equiv (\vec{u} \| \vec{v}, \vec{\mathtt{a}} \| \vec{\mathtt{b}})Z$$
$$(\vec{u},\vec{\mathtt{a}})Z \equiv Z \text{ if } \vec{u} \cap \mathrm{fnv}(Z) = \vec{\mathtt{a}} \cap \mathrm{fnv}(Z) = \varnothing$$

D Proofs

D.1 Proof of Theorem 5.1 (Unambiguous Culpability)

Since session s is not terminated, for some $\mathtt{A}, \mathtt{B}, \mathtt{e}, c, c'$ we must have:

$$T \equiv \mathtt{A}\left\langle \mathtt{B!e}; c \oplus c' \right\rangle \mid T' \quad \text{or} \quad T \equiv \mathtt{A}\left\langle \mathtt{B?e}; c + c' \right\rangle \mid (\mathtt{BA}) : \mathtt{e} \cdot \rho \mid T'$$

and therefore $T \xrightarrow{\mathtt{A} \leftrightharpoons \mathtt{B:e}}$. Notice that, using CO$_2$ congruence rule $\mathtt{A}[\mathbf{0}] \equiv \mathbf{0}$, a participant $\mathtt{A}$ (with an empty process) can be added to S even if it does not occur firsthand. Therefore, $S \equiv (\vec{u},\vec{\mathtt{c}})\,(\mathtt{A}[P] \mid s[T] \mid S_1) \mid S_2$, as required by Definition 4.1.

D.2 Proof of Theorem 5.3 (Exculpation)

By Definition 4.9, we have that for any reduction of S_0, $\mathtt{A}$ is *ready*. By assumption, $\mathtt{A}$ is culpable in S, so we must have $T \xrightarrow{\mathtt{A} \leftrightharpoons \mathtt{B:e}}$, and, by Definition 4.6, we must have $(\mathtt{B}, \mathtt{e}) \in \mathrm{WPRS}^{s}_{\mathtt{A}}(S_1)$. By Definition 4.5, we have

$$\exists S' : S_1 \xrightarrow{\neq (\mathtt{A} :\, \mathsf{do}^{s})}{}^{*}\, S' \wedge (\mathtt{B}, \mathtt{e}) \in \mathrm{PRS}^{s}_{\mathtt{A}}(S')$$

By Definition 4.4, we must have that $\mathtt{A}[\mathsf{do}^{s}_{\mathtt{B}}\,\mathtt{e}\,.\,P + P' \mid Q]$ is a participant of S'.[3] Thus, we just have to show that $\mathtt{A}[P]$ in S_0 may reduce to such a prefixed process via nonblocking actions (τ or $\mathsf{tell}_{\_}\!\downarrow_{\_}\ \_$) *only*. By contradiction, if it was the case that, in the process belonging to $\mathtt{A}$, there was a blocking action to be necessarily executed before $\mathsf{do}^{s}_{\mathtt{B}}\,\mathtt{e}$, then one could easily find a system in which $\mathtt{A}$ is not ready (i.e., $\mathtt{A}$ would be waiting for this action to be dealt with by another participant) — thus contradicting the assumption that $\mathtt{A}$ is honest.

D.3 Proof of Theorem 5.4 (Fidelity)

We note that, since S is initial, each contract in T must have been advertised along the reductions of S by one of its honest participants. The direction ($\Rightarrow$) follows directly from the semantics of CO$_2$ (i.e., rule [DO]). The direction ($\Leftarrow$) follows from Theorem 5.3: indeed, $T \xrightarrow{\mathtt{A} \leftrightharpoons \mathtt{B:e}}$ implies that $\mathtt{A}$ is culpable in session s, and by Theorem 5.3, $\mathtt{A}$ is able to exculpate itself — possibly after some nonblocking actions (i.e., no actions on session s).

---

[3] We abstract from possible choice or variable restrictions, without loss of generality.

D.4 Proof of Theorem 5.5 (Global Progress)

By contradiction. From [15], we know that T is deadlock free, thus if the CO_2 system cannot make further reductions, it must be because one of the participant `A` cannot meet its obligations. This is a contradiction with Lemma 5.3 since all participants are honest in S_0.