Notes on contract-oriented computing

Massimo Bartoletti Tiziana Cimoli Alceste Scalas
Università degli Studi di Cagliari, Italy
Department of Mathematics and Informatics
{ bart , t.cimoli , alceste.scalas } @ unica.it

January 9, 2014

Abstract

We present a theory of contract-oriented computing. *Contracts* are multi-player concurrent games, the plays of which are defined by event structures. A participant *agrees* on a contract if she has a strategy to reach her objectives (or make another participant chargeable for a violation), whatever the moves of her adversaries. A participant is *protected* by a contract when she has a strategy to defend herself in all possible contexts, even in those where she has not reached an agreement. Systems of contracting participants are modelled using the CO_2 calculus. Its primitives allow for advertising contracts, creating new sessions upon contractual agreement, and interacting on those sessions according to contractual obligations.

Contents

1 Contracts **1**
1.1 Event structures . . . 2
1.2 An event-based model of contracts . . . 2
1.3 Agreements . . . 4
1.4 Protection . . . 5

2 CO_2: a calculus of contracting processes **6**
2.1 Syntax . . . 6
2.2 Semantics . . . 7
2.3 Honesty . . . 8
2.4 Examples . . . 8

3 A mini-project **10**

1 Contracts

This section briefly reviews the contract model presented in [1]. There, contracts are modelled as concurrent systems, enriched with a notion of *obligation* (what I must do in a given state) and *objective* (what I expect to obtain in a given state). Event structures (ES) are one of the classical models for concurrency [7]. Notwithstanding the variety of formalisations, ES are at least equipped with an *enabling* relation modelling causality (usually written $\vdash$), and another relation modelling non-determinism (usually written $\#$). ES can provide a basic semantic model for contractual clauses, by interpreting the enabling $\{b\} \vdash a$ as: "I am obliged to do a *after* you have done b".

1.1 Event structures

Assume an enumerable universe of *events*, ranged over by $a, b, e, \ldots$. For a set of events X, the predicate $CF(X)$ is true iff X is *conflict-free*, i.e. $CF(X) \triangleq (\forall e, e' \in X : \neg(e \# e'))$.

Definition 1 (Event structure [7]) *An event structure $\mathcal{E}$ is a triple $\langle E, \#, \vdash \rangle$, where:*

- *E is a set of events,*
- *$\# \subseteq E \times E$ is an irreflexive and symmetric* conflict *relation,*
- *$\vdash \subseteq \{X \subseteq_{fin} E \mid CF(X)\} \times E$ is the* enabling *relation, which is* saturated*, i.e. $\forall X \subseteq Y \subseteq_{fin} E.\ X \vdash e \wedge CF(Y) \implies Y \vdash e$.*

An ES is *finite* when E is finite; it is *conflict-free* when the conflict relation is empty. We shall often use the following shorthands: $X \vdash Y$ for $\forall e \in Y.\ X \vdash e$, $a \vdash b$ for $\{a\} \vdash b$, and $\vdash e$ for $\emptyset \vdash e$.

For a sequence $\sigma = \langle e_0\, e_1 \cdots \rangle$ in E (possibly infinite), we write $\overline{\sigma}$ for the set of elements in σ; we write σ_i for the subsequence $\langle e_0 \cdots e_{i-1} \rangle$. If $\sigma = \langle e_0 \cdots e_n \rangle$, we write $\sigma\, e$ for the sequence $\langle e_0 \cdots e_n\, e \rangle$. The empty sequence is denoted by ε. For a set S, we denote with S^* the set of finite sequences over S, and with S^∞ the set of finite and infinite sequences over S.

Definition 2 (LTS of an ES) *For an* ES *$\mathcal{E}$, the labelled transition system $\mathrm{LTS}_{\mathcal{E}} = \langle \wp_{fin}(E), E, \rightarrow_{\mathcal{E}} \rangle$ is defined as follows:*

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

Definition 3 *For two* ES *$\mathcal{E}, \mathcal{E}'$, we define $\mathcal{E} \sqcup \mathcal{E}'$ as the pointwise union of $\mathcal{E}, \mathcal{E}'$.*

1.2 An event-based model of contracts

A contract (Def. 4) specifies the obligations and the objectives of a set of participants. The atomic entities of a contract are the *events*, which are uniquely associated to participants through a labelling π. Obligations are modelled as an event structure. Intuitively, an enabling $X \vdash e$ models the fact that, if all the events in X have happened, then e is an obligation for $\pi(e)$. Such obligation may be discharged only by performing e, or 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. 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 4 (Contract) *A contract $\mathcal{C}$ is a 4-tuple $\langle \mathcal{E}, \mathcal{A}, \pi, \Phi \rangle$, where:*

- *$\mathcal{E} = \langle E, \#, \vdash \rangle$ is an event structure;*
- *$\mathcal{A}$ is a set of* participants *(ranged over by $\mathsf{A}, \mathsf{B}, \ldots$);*
- *$\pi : E \to \mathcal{A}$ associates each event with a participant;*
- *$\Phi : \mathcal{A} \rightharpoonup E^\infty \to \{-1, 0, 1\}$ associates each participant and trace with a* payoff*.*

Hereafter, we shall assume that contracts respect two basic requirements. For all $X \vdash e$ in $\mathcal{E}$, we ask that $\Phi(\pi(e)) \neq \bot$. Notice that Φ is a partial function (from $\mathcal{A}$ to functions), hence a contract does not need to define payoffs for all the participants in $\mathcal{A}$ (typically, when A advertises her contract, she will not speculate about the objectives of B). The above constraint asks that if a contract defines some obligations for A, then A must also declare in $\mathcal{C}$ her payoffs.

Example 5 *Suppose there are two kids who want to play together. Alice has a toy airplane, while Bob 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 without conditions. 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 event structures:*

$$\mathcal{E}_{\mathsf{A}} : \ \{b\} \vdash a \qquad\qquad \mathcal{E}_{\mathsf{B}} : \ \emptyset \vdash b$$

The objectives of the two kids are modelled by the functions Φ_{A} (which establishes Alice's payoff) and Φ_{B} (for Bob). 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:

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

Summing up, the contracts of Alice and Bob are $\mathcal{C}_{\mathsf{A}} = \langle \mathcal{E}_{\mathsf{A}}, \mathcal{A}, \pi, \Phi_{\mathsf{A}} \rangle$ and $\mathcal{C}_{\mathsf{B}} = \langle \mathcal{E}_{\mathsf{B}}, \mathcal{A}, \pi, \Phi_{\mathsf{B}} \rangle$, respectively, where $\mathcal{A} = \{\mathsf{A}, \mathsf{B}\}$, $\pi(a) = \mathsf{A}$, and $\pi(b) = \mathsf{B}$.

Given two contracts $\mathcal{C}, \mathcal{C}'$, we denote with $\mathcal{C} \mid \mathcal{C}'$ their composition. If $\mathcal{C}'$ is the contract written by an adversary of $\mathcal{C}$, then a naïve composition of the two contracts could easily lead to an attack, e.g. when Mallory's contract says that Alice is obliged to give him her airplane. To prevent from such kinds of attacks, contract composition is a partial operation. We do *not* compose contracts which assign payoffs to the same participant, neither those which disagree on the association between events and participants.

Definition 6 (Composition of compatible contracts) *Two contracts $\mathcal{C} = \langle \mathcal{E}, \mathcal{A}, \pi, \Phi \rangle$ and $\mathcal{C}' = \langle \mathcal{E}', \mathcal{A}', \pi', \Phi' \rangle$ are* compatible *whenever:*

$$\forall e \in \mathcal{E} \cap \mathcal{E}'. \ e = e' \implies \pi(e) = \pi'(e) \tag{1}$$

$$\forall \mathsf{A} \in \mathcal{A} \cup \mathcal{A}'. \ \Phi(\mathsf{A}) = \bot \ \ \vee \ \ \Phi'(\mathsf{A}) = \bot \tag{2}$$

If $\mathcal{C}$, $\mathcal{C}'$ are compatible, we define their composition as:

$$\mathcal{C} \mid \mathcal{C}' = \langle \mathcal{E} \sqcup \mathcal{E}', \mathcal{A} \cup \mathcal{A}', \pi \sqcup \pi', \Phi \sqcup \Phi' \rangle$$

Two contracts which both assign obligations to A are not compatible.

Lemma 7 *If $\mathcal{C} = \langle \mathcal{E}, \mathcal{A}, \pi, \Phi \rangle$ and $\mathcal{C}' = \langle \mathcal{E}', \mathcal{A}', \pi', \Phi' \rangle$ are compatible, then:*

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

Example 8 *The contracts $\mathcal{C}_{\mathsf{A}}$ and $\mathcal{C}_{\mathsf{B}}$ in Ex. 5 are compatible, and their composition is the contract $\mathcal{C} = \mathcal{C}_{\mathsf{A}} \mid \mathcal{C}_{\mathsf{B}} = \langle \mathcal{E}, \mathcal{A}, \pi, \Phi \rangle$ defined as follows:*

$$\begin{array}{ll} \mathcal{E} : & \{b\} \vdash a, \ \ \emptyset \vdash b \\ \mathcal{A} : & \{\mathsf{A}, \mathsf{B}\} \\ \pi : & \pi(a) = \mathsf{A}, \ \ \pi(b) = \mathsf{B} \end{array} \qquad\qquad \Phi\mathsf{P} \ = \ \begin{cases} \Phi_{\mathsf{A}}\mathsf{A} & \text{if } \mathsf{P} = \mathsf{A} \\ \Phi_{\mathsf{B}}\mathsf{B} & \text{if } \mathsf{P} = \mathsf{B} \end{cases}$$

Definition 9 *For all contracts $\mathcal{C} = \langle \mathcal{E}, \mathcal{A}, \pi, \Phi \rangle$, we define the set $\Pi(\mathcal{C})$ of participants declaring some obligations in $\mathcal{C}$ as follows:*

$$\Pi(\mathcal{C}) \ = \ \{\pi(e) \mid X \vdash e \in \mathcal{E}\}$$

Example 10 *For the contracts in Ex. 5, we have $\Pi(\mathcal{C}_{\mathsf{A}}) = \{\mathsf{A}\}$, and $\Pi(\mathcal{C}_{\mathsf{B}}) = \{\mathsf{B}\}$.*

1.3 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 attackers. This does not mean that Alice will always succeed in all interactions: in case Bob is dishonest, we do not assume that an external authority (e.g. Bob's mother) will lend the bike to Alice. We intend that Alice agrees on a contract where, in all the interactions where she does not succeed, then some other participant must be found dishonest. That is, we consider Alice satisfied if she can blame another participant. In real-world applications, a judge may provide compensations to Alice, or impose a punishment to the participant who has violated the contract. Here, we shall not explicitly model the judge, and we shall focus instead on how to formalise the agreement property.

We interpret a contract $\mathcal{C} = \langle \mathcal{E}, \mathcal{A}, \pi, \Phi \rangle$ as a nonzero-sum concurrent multi-player game. The game involves the players in $\mathcal{A}$ concurrently performing events in order to reach the objectives defined by Φ. A *play* of $\mathcal{C}$ is a (finite or infinite) sequence of events of $\mathcal{E}$. We postulate that the permitted moves after a (finite) sequence of steps σ are exactly the events enabled by $\mathcal{E}$ in $\overline{\sigma}$, i.e. e is permitted in σ iff $\overline{\sigma} \xrightarrow{e}_{\mathcal{E}}$. A *strategy* Σ for A is a function which associates to each finite play σ a set of events of A (possibly empty), such that if $e \in \Sigma(\sigma)$ then σe is still a play. A play $\sigma = \langle e_0\, e_1 \cdots \rangle$ *conforms* to a strategy Σ for A if, for all $i \geq 0$, if $e_i \in \pi^{-1}(\mathsf{A})$, then $e_i \in \Sigma(\sigma_i)$.

As usual in concurrency, we shall only consider those *fair* plays where an event permanently 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) from keeping her promise.

Definition 11 (Fair play) *A play $\sigma = \langle e_0\, e_1 \cdots \rangle$ is* fair *w.r.t. strategy Σ iff:*

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

Our notion of agreement takes into account whether participants behave honestly in their plays. Informally, a participant is *innocent* in a play if she always keeps the promises made. An innocent participant has no persistently enabled events, i.e. all her enabled events are either performed or conflicted.

Definition 12 (Innocence) *We say that A is* innocent *in σ iff:*

$$\forall i \geq 0.\ \forall e \in \pi^{-1}(\mathsf{A}).\ \big(\overline{\sigma_i} \xrightarrow{e}_{\mathcal{E}} \implies \exists j \geq i.\ e_j \# e \ \vee\ e_j = e\big)$$

If A is not innocent in σ, then we say she is culpable.

We now define when a participant *wins* in a play. If A is culpable, then she loses. If A is innocent, but some other participant is culpable, then A wins. Otherwise, if all participants are innocent, then A wins if she has a positive payoff in the play. This is formalised as the function $\mathcal{W}$ in Def. 13 below.

Definition 13 (Winning play) *Define $\mathcal{W} : \mathcal{A} \rightharpoonup E^\infty \to \{1, 0, -1\}$ 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. $\mathcal{W}\mathsf{A}\sigma < 0$).*

We can now define when a participant *agrees* on a contract. Intuitively, A is happy to participate in an interaction regulated by contract $\mathcal{C}$ when she has a strategy Σ which allows her to win in all fair plays conform to Σ. More formally, we say that Σ is *winning* (resp. *losing*) for A iff A wins (resp. loses) in every fair play which conforms to Σ.

Definition 14 (Agreement) *A participant* A agrees *on a contract* $\mathcal{C}$ *if and only if* A *has a winning strategy in* $\mathcal{C}$. *A contract* $\mathcal{C}$ admits an agreement *whenever all the involved participants agree on* $\mathcal{C}$.

Example 15 *The contract* $\mathcal{C}$ *of Ex. 8 admits an agreement. The winning strategies for* A *and* B *are, respectively:*

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

For A*, the only fair plays conform to* Σ_{A} *are* ε *and* $\langle b\, a\rangle$. B *is culpable in* ε*, while in* $\langle b\, a\rangle$ *the payoff of* A *is positive. For* B*, the only fair plays conform to* Σ_{B} *are* $\langle b\rangle$ *and* $\langle b\, a\rangle$. A *is culpable in* $\langle b\rangle$*, while in* $\langle b\, a\rangle$ *the payoff of* B *is positive.*

1.4 Protection

In contract-oriented interactions [4], mutually distrusted 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 such contracts. When a participant agrees on a contract, she is guaranteed that — even in the presence of malicious participants — no interaction driven by the contract will ever go wrong. At worst, if A does not reach her objectives, then some other participant will be found culpable of an infringement.

This model of interaction works fine under the hypothesis that contract brokers are honest, i.e. they never establish a session in the absence of an agreement among all the participants. Suppose 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.

Formally, a contract $\mathcal{C}_{\mathsf{A}}$ *protects* A if, whatever contract $\mathcal{C}$ is composed with $\mathcal{C}_{\mathsf{A}}$, A has a way to non-lose in the composed contract.

Definition 16 (Protection) *A contract* $\mathcal{C}_{\mathsf{A}}$ protects *participant* A *if and only if, for all contracts* $\mathcal{C}$ *compatible with* $\mathcal{C}_{\mathsf{A}}$*,* A *has a non-losing strategy in* $\mathcal{C}_{\mathsf{A}} \mid \mathcal{C}$.

Notice 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 is to declare nonnegative payoffs for all σ. Less trivially, the following example shows a contract with possible negative payoffs which still offers protection.

Example 17 *The contract* $\mathcal{C}_{\mathsf{B}}$ *of Ex. 5 does* not *protect Bob. To prove that, consider e.g. the attacker contract* $\mathcal{C}' = \langle \mathcal{E}', \mathcal{A}, \pi, \Phi_{\mathcal{C}'}\rangle$*, where* $\mathcal{A}$ *and* π *are as in Ex. 5, while we define* $\mathcal{E}'$ *with no enablings, and* $\Phi_{\mathcal{C}'}$ *is immaterial except for being undefined on* B

(otherwise $\mathcal{C}'$ and $\mathcal{C}_\mathsf{B}$ are not compatible). 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 fair play $\sigma = \langle b\rangle$ we have that $\Phi\mathsf{B}\sigma = -1$, but no participant is culpable in σ. By Def. 16,* B *is not protected by $\mathcal{C}_\mathsf{B}$.*

On the other hand, the contract $\mathcal{C}_\mathsf{A}$ protects Alice. To show that, consider a contract $\mathcal{C}$ compatible 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 fair play in $\mathcal{C} \mid \mathcal{C}_\mathsf{A}$ conforming to Σ_A. There are two cases:

- *$b \in \overline{\sigma}$. Then, since σ is fair, by definition of Σ_A there must exist i such that $a \in \overline{\sigma_i}$, and so* A *is innocent in σ. Furthermore, we have that $\Phi\mathsf{A}\sigma = 1$.*
- *$b \notin \overline{\sigma}$. By definition of $\mathcal{C}_\mathsf{A}$,* 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*.*

2 CO_2: a calculus of contracting processes

We now embed the contracts introduced in previous section in the process calculus CO_2 [2, 3, 4, 5]. Let $\mathcal{V}$ and $\mathcal{N}_S$ be two disjoint countably infinite sets of *session variables* (ranged over by $x, y, \ldots$) and *session names* (ranged over by $s, t, \ldots$). Let $u, v, \ldots$ range over $\mathcal{V} \cup \mathcal{N}_S$.

2.1 Syntax

Definition 18 *The abstract syntax of CO_2 is given by the following productions:*

$$\begin{array}{rcl} P & ::= & \sum_i \pi_i.P_i \mid P|P \mid (\vec{u})P \mid X(\vec{u}) \\ \pi & ::= & \tau \mid \mathsf{tell}_\mathsf{A}{\downarrow_u}\,\mathcal{C} \mid \mathsf{fuse} \mid \mathsf{do}_u\,a \mid \mathsf{ask}_u\,\phi \\ K & ::= & {\downarrow_u}\,\mathsf{A}\,\mathit{says}\,\mathcal{C} \mid K|K \\ S & ::= & \mathbf{0} \mid \mathsf{A}[P] \mid \mathsf{A}[K] \mid s[\mathcal{C} : \sigma] \mid S|S \mid (\vec{u})S \end{array}$$

The only binder for session variables and names is the delimitation $(\vec{u})$, both in systems and processes. Free variables/names are defined accordingly, and they are denoted by $\mathrm{fv}(\_)$ and $\mathrm{fn}(\_)$. A system or a process is *closed* when it has no free variables.

Systems are the parallel composition of *participants* $\mathsf{A}[P]$, *latent contracts* (collected by A) $\mathsf{A}[K]$, and *sessions* $s[\mathcal{C} : \sigma]$.

A *latent contract* ${\downarrow_x}\,\mathsf{A}$ *says* $\mathcal{C}$ represents a contract $\mathcal{C}$ (advertised by A) which has not been stipulated yet; upon stipulation, x will be instantiated to a fresh session name. We allow prefix-guarded finite sums of processes, and write $\pi_1.P_1 + \pi_2.P_2$ for $\sum_{i=1,2} \pi_i.P_i$, and $\mathbf{0}$ for $\sum_\emptyset P$. Recursion is allowed only for processes; for this we stipulate that each process identifier X has a unique defining equation $X(u_1, \ldots, u_j) \stackrel{\text{def}}{=} P$ such that $\mathrm{fv}(P) \subseteq \{u_1, \ldots, u_j\} \subseteq \mathcal{V}$ and each occurrence of process identifiers in P is prefix-guarded. We shall take the liberty of omitting the arguments of $X(\vec{u})$ when they are clear from the context.

commutative monoidal laws for $\mid$ on processes and systems

$$Z \mid (u)Z' \equiv (u)(Z \mid Z') \quad \text{if } u \notin \mathrm{fv}(Z) \cup \mathrm{fn}(Z)$$

$$(u)Z \equiv Z \qquad \text{if } u \notin \mathrm{fv}(Z) \cup \mathrm{fn}(Z)$$

$$(u)(v)Z \equiv (v)(u)Z \qquad \mathsf{A}[(v)P] \equiv (v)\mathsf{A}[P] \qquad \mathsf{A}[K] \mid \mathsf{A}[K'] \equiv \mathsf{A}[K \mid K']$$

Figure 1: Structural equivalence for CO_2 (Z, Z' range over systems or processes)

$$\mathsf{A}[\tau.P + P' \mid Q] \to \mathsf{A}[P \mid Q] \qquad \text{[Tau]}$$

$$\mathsf{A}[\mathsf{tell}_{\mathsf{B}} \downarrow_x c.P + P' \mid Q] \to \mathsf{A}[P \mid Q] \;\mid\; \mathsf{B}[\downarrow_x \mathsf{A}\ \mathit{says}\ c] \qquad \text{[Tell]}$$

$$\frac{\begin{array}{c} K = \downarrow_{x_1} \mathsf{A}_1\ \mathit{says}\ \mathcal{C}_1 \mid \ldots \mid \downarrow_{x_n} \mathsf{A}_n\ \mathit{says}\ \mathcal{C}_n \qquad \forall i \in 1..n\,.\,\Pi(\mathcal{C}_i) = \{\mathsf{A}_i\} \\ \mathcal{C} = \mathcal{C}_1 \mid \ldots \mid \mathcal{C}_n \text{ admits agreement} \qquad \rho = \{^s/_{x_1,\ldots,x_n}\} \qquad s \text{ fresh} \end{array}}{(x_1 \cdots x_n)(\mathsf{A}[\mathsf{fuse}.P + P' \mid Q] \mid \mathsf{A}[K] \mid S) \to (s)(\mathsf{A}[P \mid Q]\rho \;\mid\; s[\mathcal{C} : \langle\rangle] \;\mid\; S\rho)} \qquad \text{[Fuse]}$$

$$\frac{\mathcal{C} = \langle \mathcal{E}, \mathcal{A}, \pi, \Phi \rangle \qquad \pi(a) = \mathsf{A} \qquad \sigma \xrightarrow{a} \sigma a}{s[\mathcal{C} : \sigma] \;\mid\; \mathsf{A}[\mathsf{do}_s\ a.P + P' \mid Q] \to s[\mathcal{C} : \sigma a] \;\mid\; \mathsf{A}[P \mid Q]} \qquad \text{[Do]}$$

$$\frac{\mathcal{C} : \sigma \vdash \phi}{\mathsf{A}[\mathsf{ask}_s\ \phi.P + P' \mid Q] \mid s[\mathcal{C} : \sigma] \to \mathsf{A}[P \mid Q] \mid s[\mathcal{C} : \sigma]} \qquad \text{[Ask]}$$

$$\frac{X(\vec{u}) \stackrel{\text{def}}{=} P \qquad \mathsf{A}[P\{^{\vec{v}}/_{\vec{u}}\} \mid Q] \mid S \to S'}{\mathsf{A}[X(\vec{v}) \mid Q] \mid S \to S'} \qquad \text{[Def]}$$

$$\frac{S \to S'}{S \mid S'' \to S' \mid S''} \qquad \text{[Par]}$$

$$\frac{S \to S'}{(u)S \to (u)S'} \qquad \text{[Del]}$$

Figure 2: Reduction semantics of CO_2

Prefixes include silent action τ, contract advertisement $\mathsf{tell}_{\mathsf{A}} \downarrow_u \mathcal{C}$, contract stipulation fuse, action execution $\mathsf{do}_u\ a$, and contract query $\mathsf{ask}_u\ \phi$. In each prefix $\pi \neq \tau$, the name/variable u refers to the target session involved in the execution of π. We omit trailing occurrences of $\mathbf{0}$.

Note that participants can only contain latent contracts, while sessions can only contain contracts, constructed from latent contracts upon reaching agreements.

2.2 Semantics

The semantics of CO_2 is formalised by a reduction relation $\to$ on systems, which relies on the structural congruence defined in Fig. 1.

Definition 19 *The relation $\to$ is the smallest relation closed under the rules of Fig. 2, defined over systems up to structural equivalence, as defined in Fig. 1.*

We now briefly discuss the rules in Fig. 2.

- [Tau] simply fires a τ prefix.
- [Tell] advertises a latent contract $\downarrow_x \mathsf{A}$ *says* $\mathcal{C}$, by collecting it under a process B (which is supposed to play the role of a contract broker).
- [Fuse] creates a new session s when an agreement exists among the latent contracts K advertised to A; technically, s is shared among all the involved participants $\mathsf{A}_1, \ldots, \mathsf{A}_n$ through the substitution ρ, which replaces the delimited variables $x_1, \ldots, x_n$. As a safety measure, the rule requires that the contract $\mathcal{C}_i$ of each participant A_i only associates events to A_i itself. The state of the new session s is composed by a contract $\mathcal{C} = \mathcal{C}_1 \mid \ldots \mid \mathcal{C}_n$, and an empty play $\langle\rangle$.
- [Do] allows a participant A to fulfill its obligations according to the contract $\mathcal{C}$ in session s, by performing some enabled event a. The session state evolves accordingly: a is added to the play σ. Note that the events not enabled in σ cannot be fired (that is, only obliged events are permitted).
- [Ask] checks if a condition ϕ holds in a session. The actual nature of ϕ is almost immaterial in these notes: in the examples below we shall just write ϕ in prose. More formally, one can assume, for instance, that ϕ is a formula in an LTL logic [6].
- The last three rules are standard.

2.3 Honesty

We now define when a participant is *honest*. Intuitively, honest participants always respect the contracts they advertise. This notion is crucial in contract-oriented systems, since honest participants will never be liable in case of misbehaviours.

More precisely, a participant A is honest when she is never persistently culpable in any session she may be engaged in — even in systems populated by adversaries who play to cheat her. Thus, if a system S contains a session s with some enabled event a pertaining to A, then A must either fire aa, or conflict it.

Before introducing honesty, whe define when a participant is innocent (or culpable) in a system.

Definition 20 (Innocence in $\mathbf{CO_2}$) *A participant* A *is* innocent in a system S *iff, whenever* $S \equiv (\vec{u})\,(s[\mathcal{C} : \sigma] \mid S')$*, we have that* A *is innocent in* σ*. If* A *is not innocent in* S*, then we say she is* culpable in S*.*

Notice that, in particular, A is innocent in a system S when S does not contain sessions where A is involved in. Indeed, in these systems no events of A are ever enabled.

Definition 21 (Honesty) $\mathsf{A}[P]$ *(with* P *closed) is* honest *iff for all* A*-free systems* S*,*

$$\mathsf{A}[P] \mid S \to^* S' \quad \Longrightarrow \quad \exists S'' \;.\; S' \to^* S'' \text{ and } \mathsf{A} \text{ is innocent in } S''$$

Intuitively, Def. 21 requires that if we insert $\mathsf{A}[P]$ in a context S not containing latent or stipulated contracts of A, then whenever some system S' is reached, there always exist a reachable S'' where A is innocent. Hence, even when A is culpable in S', she can eventually exculpate. Note that since there is no assumption on what other participants may (or may not) do in S', A must be able to exsculpate "on her own", without expecting others to fulfil their contracts.

2.4 Examples

The rest of this section is devoted to a few examples that highlight how contracts can be used in CO_2.

Example 22 *Recall the contracts* $\mathcal{C}_\mathsf{A}$ *and* $\mathcal{C}_\mathsf{B}$ *from Ex. 5. A possible specification of the processes of* A *and* B *can be the following, where* M *(for "Mom") is a special participant acting as a contract broker.*

$$\begin{array}{rcl} P_\mathsf{A} & = & (x)\ \mathsf{tell}_\mathsf{M}{\downarrow_x}\ \mathcal{C}_\mathsf{A}.\ \mathsf{do}_x\ a \\ P_\mathsf{B} & = & (y)\ \mathsf{tell}_\mathsf{M}{\downarrow_y}\ \mathcal{C}_\mathsf{B}.\ \mathsf{do}_y\ b \end{array}$$

We have the following computation of the system $S = \mathsf{A}[P_\mathsf{A}] \mid \mathsf{B}[P_\mathsf{B}] \mid \mathsf{M}[\mathsf{fuse}]$*:*

$$\begin{array}{rl} S \to & (x)\big(\mathsf{A}[\mathsf{do}_x\ a] \mid \mathsf{M}[\mathsf{A}\ \mathit{says}\ {\downarrow_x}\ \mathcal{C}_\mathsf{A}]\big) \mid \mathsf{B}[P_\mathsf{B}] \mid \mathsf{M}[\mathsf{fuse}] \\ \to & (x)\big(\mathsf{A}[\mathsf{do}_x\ a] \mid \mathsf{M}[\mathsf{A}\ \mathit{says}\ {\downarrow_x}\ \mathcal{C}_\mathsf{A}]\big) \mid (y)\big(\mathsf{B}[\mathsf{do}_y\ b] \mid \mathsf{M}[\mathsf{B}\ \mathit{says}\ {\downarrow_y}\ \mathcal{C}_\mathsf{B}]\big) \mid \mathsf{M}[\mathsf{fuse}] \\ \equiv & (x,y)\big(\mathsf{A}[\mathsf{do}_x\ a] \mid \mathsf{B}[\mathsf{do}_y\ b] \mid \mathsf{M}[\mathsf{A}\ \mathit{says}\ {\downarrow_x}\ \mathcal{C}_\mathsf{A} \mid \mathsf{B}\ \mathit{says}\ {\downarrow_y}\ \mathcal{C}_\mathsf{B}] \mid \mathsf{M}[\mathsf{fuse}]\big) \end{array}$$

We know from Ex. 15 that $\mathcal{C} = \mathcal{C}_\mathsf{A} \mid \mathcal{C}_\mathsf{B}$ *admits an agreement, hence by rule* [Fuse]*:*

$$\to (s)\big(\mathsf{A}[\mathsf{do}_s\ a] \mid \mathsf{B}[\mathsf{do}_s\ b] \mid s[\mathcal{C} : \varepsilon]\big)$$

Here we have that $\varepsilon \xrightarrow{b}$*, and* $\varepsilon \not\xrightarrow{a}$*, hence by rule* [Do]*, the only possible transition is:*

$$\to (s)\big(\mathsf{A}[\mathsf{do}_s\ a] \mid \mathsf{B}[\mathbf{0}] \mid s[\mathcal{C} : \langle b \rangle]\big)$$

Now we have that $\langle b \rangle \xrightarrow{a}$ *hence by rule* [Do]*:*

$$\to (s)\big(\mathsf{A}[\mathbf{0}] \mid \mathsf{B}[\mathbf{0}] \mid s[\mathcal{C} : \langle ba \rangle]\big)$$

In conclusion, Alice and Bob have reached their goals in the above computation.

It is also possible to notice that $\mathsf{A}[P_A]$ *and* $\mathsf{A}[P_B]$ *are both honest.*

Example 23 *Let us consider an on-line store (participant* A*), which sells apples (a) and bottles of an expensive italian Brunello wine (b). Selling apples is quite easy: once an order is placed,* A *accepts it (with the feedback ok) and waits for a payment (pay) before shipping the goods (ship-a). However, if expensive bottles of Brunello are ordered, the store is entitled to either decline the order (by answering no), or accept it (and, as above, ship the item after the payment, with ship-b).*

The store contract can be modeled as $\mathcal{C}_\mathsf{A} = \langle \mathcal{E}_\mathsf{A}, \mathcal{A}, \pi, \Phi_\mathsf{A} \rangle$*, where:*

$$\begin{array}{rcl} \mathcal{E}_\mathsf{A} & : & \begin{cases} a \vdash \mathit{ok},\ b \vdash \mathit{ok},\ b \vdash \mathit{no}, \quad a \# b,\ \mathit{ok} \# \mathit{no} \\ \{a, \mathit{pay}\} \vdash \mathit{ship\text{-}a},\quad \{b, \mathit{pay}\} \vdash \mathit{ship\text{-}b} \end{cases} \\ \mathcal{A} & = & \{\mathsf{A}, \mathsf{B}\} \\ \pi & : & a \mapsto \mathsf{B},\ b \mapsto \mathsf{B},\ \mathit{ok} \mapsto \mathsf{A},\ \mathit{no} \mapsto \mathsf{A},\ \mathit{pay} \mapsto \mathsf{B},\ \mathit{ship\text{-}a} \mapsto \mathsf{A},\ \mathit{ship\text{-}b} \mapsto \mathsf{A} \\ \Phi_\mathsf{A}\mathsf{A}\sigma & = & \begin{cases} 1 & \text{if } \mathit{pay} \in \overline{\sigma} \text{ or } (\mathit{ship\text{-}a} \notin \overline{\sigma} \text{ and } \mathit{ship\text{-}b} \notin \overline{\sigma}) \\ 0 & \text{otherwise} \end{cases} \end{array}$$

A buyer B *who only wants to buy Brunello can advertise the contract* $\mathcal{C}_\mathsf{B} = \langle \mathcal{E}_\mathsf{B}, \mathcal{A}, \pi, \Phi_\mathsf{B} \rangle$*, where* $\mathcal{A}$ *and* π *are as above, and:*

$$\begin{array}{rcl} \mathcal{E}_\mathsf{A} & : & \vdash b,\ \mathit{ok} \vdash \mathit{pay} \\ \Phi_\mathsf{B}\mathsf{B}\sigma & = & \begin{cases} 1 & \text{if } \mathit{pay} \in \overline{\sigma} \implies \mathit{ship\text{-}b} \in \overline{\sigma} \\ 0 & \text{otherwise} \end{cases} \end{array}$$

The interaction between the store A and the buyer B could be modeled with the following CO_2 system (where C is a broker):

$$\begin{array}{rcl} S & = & (x)\mathsf{A}[\mathsf{tell}_{\mathsf{C}}\downarrow_x \mathcal{C}_{\mathsf{A}} \,.\, P_{\mathsf{A}}] \;\mid\; (y)\mathsf{B}[\mathsf{tell}_{\mathsf{C}}\downarrow_y \mathcal{C}_{\mathsf{B}} \,.\, P_{\mathsf{B}}] \;\mid\; \mathsf{C}[\mathsf{fuse}] \\ P_{\mathsf{A}} & \stackrel{\text{def}}{=} & \mathsf{do}_x\ ok \,.\, (\mathsf{do}_x\ ship\text{-}a + \mathsf{do}_x\ ship\text{-}b) \;+\; \mathsf{do}_x\ no \\ P_{\mathsf{B}} & \stackrel{\text{def}}{=} & \mathsf{do}_y\ b \,.\, (\mathsf{ask}_y\ ok? \,.\, \mathsf{do}_y\ pay \;+\; \mathsf{ask}_y\ no? \,.\, \mathbf{0}) \end{array}$$

Here, both A and B are *honest*: in all their possible reductions, they are always able to perform the actions expected by their contracts. Note that the $\mathsf{do}_x\ no$ in the main choice of P_{A} can only be fired when an agreement is found, a session is fused and the corresponding *no* event is enabled — i.e., when the customer selected b.

Another possible honest implementation of P_{A} is the following, where A chooses to always refuse b orders:

$$P'_{\mathsf{A}} \;=\; \mathsf{ask}_x\ a? \,.\, \mathsf{do}_x\ ok \,.\, \mathsf{do}_x\ ship\text{-}a \;+\; \mathsf{ask}_x\ b? \,.\, \mathsf{do}_x\ no$$

3 A mini-project

Consider a travel agency A which queries in parallel an airline ticket broker F and a hotel reservation service H in order to complete the organization of a trip. The travel agency advertises a contract $\mathcal{C}_{\mathsf{A}}$, with the following obligations:

$$\mathcal{E}_{\mathsf{A}}: \quad pay \vdash hotel \qquad pay \vdash flight \qquad pay \vdash refund \qquad refund\#hotel \qquad refund\#flight$$

This models the fact that when a client pays, the agency must either provide the client with hotel and flight reservations, or refund the paid amount.

In order to provide the client with the reservations, the agency must exploit the services F and H. To do that, the agency advertises two contracts $\mathcal{C}_{\mathsf{AH}}$, $\mathcal{C}_{\mathsf{AF}}$, with the following obligations:

$$\begin{array}{lllll} \mathcal{E}_{\mathsf{AH}}: & \vdash queryH & quoteH \vdash payH & quoteH \vdash abortH & abortH\#payH \\ \mathcal{E}_{\mathsf{AF}}: & \vdash queryF & quoteF \vdash payF & quoteF \vdash abortF & abortF\#payF \end{array}$$

For instance, $\mathcal{E}_{\mathsf{H}}$ models the fact that A asks H a quotation for the hotel, and then can either accept it (*payH*) or not (*abortH*). The goal of A is to obtain from H and F the hotel/flight reservations (*bookH* and *bookF*, respectively). The agency will provide the client with *hotel* and *flight* only after both reservations have been obtained.

1. Define the payoff functions Φ_{A} and Φ_{AH} of the travel agency.
2. Write a contract $\mathcal{C}_{\mathsf{B}}$ for the client, and show that:
   (a) $\mathcal{C}_{\mathsf{A}} \mid \mathcal{C}_{\mathsf{B}}$ admits an agreement.
   (b) $\mathcal{C}_{\mathsf{A}}$ protects A.
   (c) $\mathcal{C}_{\mathsf{B}}$ does not protect B.
3. Write a contract $\mathcal{C}_{\mathsf{H}}$ for the hotel reservation service, and show that $\mathcal{C}_{\mathsf{AH}} \mid \mathcal{C}_{\mathsf{H}}$ admits an agreement.
4. Design CO_2 processes P_{A}, P_{B}, P_{H}, P_{F} for the four participants.
5. Write (step-by-step) a maximal computation of $\mathsf{A}[P_{\mathsf{A}}] \mid \mathsf{B}[P_{\mathsf{B}}] \mid \mathsf{H}[P_{\mathsf{H}}] \mid \mathsf{F}[P_{\mathsf{F}}]$.
6. Discuss what happens when one (or both) of the participants H, F are not honest.
7. Discuss what happens when extending CO_2 with the following rule:

$$\frac{K = \downarrow_{x_1} \mathsf{A}_1\ says\ \mathcal{C}_1 \mid \ldots \mid \downarrow_{x_n} \mathsf{A}_n\ says\ \mathcal{C}_n \qquad \forall i \in 1..n \,.\, \Pi(\mathcal{C}_i) = \{\mathsf{A}_i\} \qquad \mathcal{C} = \mathcal{C}_1 \mid \ldots \mid \mathcal{C}_n \qquad \rho = \{{}^{s}/_{u_1,\ldots,u_n}\} \qquad s \text{ fresh}}{(x_1 \cdots x_n)(\mathsf{A}[\mathsf{fuse}.P + P' \mid Q] \mid \mathsf{A}[K] \mid S) \to (s)(\mathsf{A}[P \mid Q]\rho \;\mid\; s[\mathcal{C} : \langle\rangle] \;\mid\; S\rho)}$$

References

[1] M. Bartoletti, T. Cimoli, and R. Zunino. A theory of agreements and protection. In *Proc. POST*, volume 7796 of *LNCS*. Springer, 2013.

[2] M. Bartoletti, A. Scalas, E. Tuosto, and R. Zunino. Honesty by typing. In *Proc. FORTE*, 2013.

[3] M. Bartoletti, E. Tuosto, and R. Zunino. Contracts in distributed systems. In *ICE*, 2011.

[4] M. Bartoletti, E. Tuosto, and R. Zunino. Contract-oriented computing in CO_2. *Scientific Annals in Computer Science*, 22(1):5–60, 2012.

[5] M. Bartoletti, E. Tuosto, and R. Zunino. On the realizability of contracts in dishonest systems. In *Proc. COORDINATION*, 2012.

[6] E. A. Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*. North-Holland Pub. Co./MIT Press, 1990.

[7] G. Winskel. Event structures. In *Advances in Petri Nets*, pages 325–392, 1986.