Rafael H. Bordini, Louise A. Dennis and Yves Lespérance (eds)
Pre-proceedings of the 7th International Workshop on Engineering Multi-Agent Systems (EMAS 2019)
13th-14th May 2019, Montreal, Canada
cgi.csc.liv.ac.uk/~lad/emas2019/accepted/EMAS2019_paper_29.pdf

On Enactability of Agent Interaction Protocols: Towards a Unified Approach

Angelo Ferrando[1], Michael Winikoff[2], Stephen Cranefield[2], Frank Dignum[3], and Viviana Mascardi[4]

[1] Liverpool University, Liverpool, United Kingdom, angelo.ferrando@liverpool.ac.uk*
[2] University of Otago, Dunedin, New Zealand, name.surname@otago.ac.nz
[3] Umeå University, Umeå, Sweden, Utrecht University, The Netherlands, CVUT Prague, Czech Republic, frank.dignum@umu.se
[4] Genova University, Genova, Italy, viviana.mascardi@unige.it

Abstract. Interactions between agents are usually designed from a global viewpoint. However, the implementation of a multi-agent interaction is distributed. This difference can introduce problems. For instance, it is possible to specify protocols from a global viewpoint that cannot be implemented as a collection of individual agents. This leads naturally to the question of whether a given (global) protocol is *enactable*. We consider this question in a powerful setting (trace expressions), considering a range of message ordering interpretations (specifying what it means to say that an interaction step occurs before another), and a range of possible constraints on the semantics of message delivery, corresponding to different properties of the underlying communication middleware.

Keywords: Agent Interaction Protocols · Enactability · Enforceability · Implementability · Realizability · Projectability · Trace Expressions

1 Introduction

In order to organise her stay in Montreal, Alice books an apartment from Bob via the online platform AIPbnb. AIPbnb policy states that owners cannot interact with each other, users can interact with owners only via the platform, and if a user finds a better solution for her accommodation, she must cancel the previous one *before* she makes a new reservation for the same dates, otherwise she will be charged for one night there. When Alice discovers that Carol rents a cheaper and larger apartment, she decides to cancel the reservation of Bob's apartment and book Carol's one. This situation can be represented by the global Agent Interaction Protocol $modifyRes = Alice \overset{Canc}{\Longrightarrow} Bob \cdot Alice \overset{Res}{\Longrightarrow} Carol$ where $a1 \overset{M}{\Longrightarrow} a2$ models the interaction between $a1$ and $a2$ to exchange message M, "$\cdot$" models interaction concatenation, and $Canc$ and Res are sent to the recipients by using the AIPbnb platform as required. Alice believes that the above protocol correctly meets AIPbnb policy, but she is charged for one night in Bob's

* Work supported by EPSRC as part of the ORCA [EP/R026173] and RAIN [EP/R026084] Robotics and AI Hubs.

apartment by AIPbnb: Carol received Alice's request before Bob received the cancellation, and this violates the policy. What went wrong is the *interpretation* of "before". To Alice, it meant that she should send $Canc$ before she sent Res, while for AIPbnb it (also) meant that Bob should receive $Canc$ before Carol received Res. This ambiguity would have had no impact on Alice if the physical *communication model* underlying AIPbnb guaranteed that between the sending and receiving stages of an interaction, nothing could happen. However, if the communication model provides weaker or no guarantees, it may happen that a message sent before another, is delivered after.

This simple example shows that enacting the intent of a global protocol without clear semantics of the meaning of "before", without guarantees from the platform implementation on message delivery order, and without hidden communications between the participants ("covert channels"), may not be possible. Many real situations are similar to this one: for example, a citizen must wait for the bank to have received (and processed) the request to add some money to a new empty account, before sending a request to move that money to another account, otherwise he can go into overdraft.

This kind of issue is not new in the field and various authors use different terms for global protocols that can be enforced by distributed participants: conformant [18], enforceable [11, 4], enactable [12], implementable [21], projectable [8, 16], realizable [22, 19]. The concept behind these names is however the same: by executing the localised versions of the protocol implemented by each participant, the global protocol behaviour is obtained, with no additional communication. We will use the term *enactability* to denote this property. However, despite the large amount of work on enactability, there is no existing work that considers both the intended *message ordering* and the *communication model* of the infrastructure in which the agents will be implemented, that recognises the need to use a *decision structure* to enforce consistent choices, and that provides an implementation for checking protocol enactability. Together, these are the innovative and original features of our contribution (a detailed discussion of related work is in Section 4).

Although it might be argued that it is desirable to have robust protocol specifications that are independent of the underlying platform implementation, we observe that robustness can make the protocol more complex, and hence harder to maintain. For example, considering again the protocol $modifyRes = Alice \overset{Canc}{\Longrightarrow} Bob \cdot Alice \overset{Res}{\Longrightarrow} Carol$, we observe that, depending on which interpretation we choose, we can have different conclusions on what to expect from the protocol implementation. This can be avoided if we add additional acknowledgement messages, which gives a more message-intensive protocol such as $modifyRes = Alice \overset{Canc}{\Longrightarrow} Bob \cdot Bob \overset{Ack}{\Longrightarrow} Alice \cdot Alice \overset{Res}{\Longrightarrow} Carol$, in which $Alice$ would not be charged erroneously. However, adding additional acknowledgement messages increases the complexity of the protocol and reduces opportunities for concurrency. We therefore prefer to take into account what the underlying implementation guarantees wth respect to communication, so that we can relax our specifications, and use as simple a protocol as possible. Additionally, a protocol that is not enactable in some platform may be enactable in some other platform. Our work is therefore relevant to both platform designers and protocol designers.

2 Background

Trace Expressions. Trace expressions [3] are a compact and expressive formalism inspired by global types [1] and then extended and exploited in different application domains [14, 2, 13]. Initially devised for runtime verification of multiagent systems, trace expressions are expressive, and can define context-sensitive languages.

A trace expression τ denotes a set of possibly infinite event traces, and is defined on top of the following operators:[5]

- ϵ (empty trace), denoting the singleton set $\{\langle\rangle\}$ containing the empty event trace $\langle\rangle$.
- M (event), denoting a singleton set $\{\langle M\rangle\}$ containing the event trace $\langle M\rangle$.
- $\tau_1 \cdot \tau_2$ (*concatenation*), denoting the set of all traces obtained by concatenating the traces of τ_1 with those of τ_2.
- $\tau_1 \wedge \tau_2$ (*intersection*), denoting the intersection of the traces of τ_1 and τ_2.
- $\tau_1 \vee \tau_2$ (*union*), denoting the union of the traces of τ_1 and τ_2.
- $\tau_1 | \tau_2$ (*shuffle*), denoting the union of the sets obtained by shuffling each trace of τ_1 with each trace of τ_2 (see [7] for a more precise definition).

Trace expressions are cyclic terms, thus they can support recursion without introducing an explicit construct.

As customary, the operational semantics of trace expressions, defined in [3], is specified by a transition relation $\delta \subseteq \mathcal{T} \times \mathcal{E} \times \mathcal{T}$, where $\mathcal{T}$ and $\mathcal{E}$ denote the sets of trace expressions and of events, respectively. We do not present all the transition rules for space constraints. They are standard ones (see e.g. [3]) that state, for example, that $\delta(ev \cdot \tau, ev, \tau)$ (the protocol whose state is modelled by $ev \cdot \tau$ can move to state τ if ev occurs), and that $\delta(\tau_1 \vee \tau_2, ev, \tau)$ if $\delta(\tau_1, ev, \tau)$ (if the protocol whose state is modelled by τ_1 can move to state τ if ev occurs, then also the protocol whose state is modelled by $\tau_1 \vee \tau_2$ can). The denotational semantics is defined as follows, Where $t_1 \bowtie t_2$ is the set of all interleavings of t_1 and t_2, and $\circ$ is concatenation over sequences:

$$
\begin{aligned}
[\![\epsilon]\!] &= \{\langle\rangle\} \\
[\![M]\!] &= \{\langle M\rangle\} \\
[\![\tau_1 \cdot \tau_2]\!] &= \{t_1 \circ t_2 \mid t_1 \in [\![\tau_1]\!] \wedge t_2 \in [\![\tau_2]\!]\} \\
[\![\tau_1 \wedge \tau_2]\!] &= [\![\tau_1]\!] \cap [\![\tau_2]\!] \\
[\![\tau_1 \vee \tau_2]\!] &= [\![\tau_1]\!] \cup [\![\tau_2]\!] \\
[\![\tau_1 | \tau_2]\!] &= \{z \mid t_1 \in [\![\tau_1]\!] \wedge t_2 \in [\![\tau_2]\!] \wedge z \in t_1 \bowtie t_2\}
\end{aligned}
$$

Events can be, in principle, of any kind. In this paper, we will limit ourselves to consider *interaction* and *message* events.

An interaction has the form $a \overset{M}{\Longrightarrow} b$ and gives information on the protocol from the global perspective, collapsing sending and receiving into a single event. We say that τ

[5] Binary operators associate from left, and are listed in decreasing order of precedence; that is, the first operator has the highest precedence. The operators "$\vee$" and "$\wedge$" are the standard notation for trace expressions.

is an *interaction protocol* if all the events therein are interactions. Interaction protocols take other names in other communities, such as Interaction Oriented Choreography [18] in the service-oriented computing community, and global type in the community working on process calculi and types [9].

Message events have the form $aM!$ (a sends M) and $bM?$ (b receives M). They model actions that one agent can execute, hence taking a local perspective. A trace expression where all events are messages will be named a *message protocol* throughout the paper. Message protocols have different names in different communities, such as Process Oriented Choreography [18] and "local type" or "session type" in the global types community [15, 24].

Communication Models. Given that in our proposal we explicitly take the communication model supported by the MAS infrastructure into account, we provide a summary of communication models based on [10]. We use CM0 to CM6 to identify them in a compact way.

CM0: Synchronous Communication. Sending and receiving are synchronised: the sender cannot send if the receiver is not ready to receive.

CM1: Realisable with Synchronous Communication (RSC). After a communication transition consisting of a send event of a message, the only possible communication transition is the receive event of this message. This asynchronous model is the closest one to synchronous communication and can be implemented with a 1-slot unique buffer shared by all agents.

CM2: FIFO n-n communication. Messages are globally ordered and are delivered in their emission order: if sending of M_1 takes place before sending of M_2, then reception of M_1 must take place before reception of M_2. This model can be implemented by means of a shared centralised object, such as unique queue.

CM3: FIFO 1-n communication. Messages from the same sender are delivered in the order in which they were sent. It can be implemented by giving each agent a unique queue where it puts its outgoing messages, with peers fetching messages from this queue.

CM4: FIFO n-1 communication. A send event is implicitly and globally ordered with regard to all other sending actions toward the same agent. This means that if agent b receives M_1 (sent by agent a) and later it receives M_2 (sent by agent c), b knows that the sending of M_1 occurred before the sending of M_2 in the global execution order, even if there is no causal path between the two sending actions. The implementation of this model can, similarly to FIFO 1-n, be done by providing each agent with a queue: messages are sent by putting them into the queue of the recipient agent.

CM5: Causal. Messages are delivered according to the causality of their emissions [17]: if a message M_1 is causally sent before a message M_2 then an agent cannot get M_2 before M_1. Implementing this model requires sharing the causality relation.

CM6: Fully Asynchronous. No order on message delivery is imposed. Messages can overtake others or be arbitrarily delayed. The implementation can be modelled by a bag.

Message Ordering. The statement "one interaction comes before another" is ambiguous, as exemplified in Section 1. This ambiguity has been recognised by some authors who suggested how to interpret message ordering when moving from the interaction

(global) level to the message (local) level. In this section we summarise and compare the proposals by Lanese *et al.* [18] and Desai and Singh [12].

To identify the interpretations, we will use the acronyms used in [12] when available, and our own acronyms otherwise. The starting point for interpreting message ordering is the interaction protocol $\tau = a \overset{M_1}{\Longrightarrow} b{\cdot}c \overset{M_2}{\Longrightarrow} d$. For the sake of clarity, we denote $aM_1!$ with $s1$, $bM_1?$ with $r1$, $cM_2!$ with $s2$, and $dM_2?$ with $r2$; we characterise the message ordering interpretations by the traces of message events that respect them.
RS: Under this message ordering interpretation the meaning of "interaction event M_1 occurs before M_2" is that M_1 is received before M_2 is sent. The set of traces that respect this model is $\{\langle s1, r1, s2, r2\rangle\}$. This interpretation is named *RS (receive before send)* in [12] and *disjoint semantics* in [18].
SS: M_1 is sent before M_2, and there are no constraints on the delivery order. The set of traces that respect this model is $\{\langle s1, r1, s2, r2\rangle, \langle s1, s2, r1, r2\rangle, \langle s1, s2, r2, r1\rangle\}$. This interpretation is named *SS (send before send)* in [12] and *sender semantics* in [18].
RR: M_1 is received before M_2, and there are no constraints on the sending order. The set of traces that respect this model is $\{\langle s1, r1, s2, r2\rangle, \langle s1, s2, r1, r2\rangle, \langle s2, s1, r1, r2\rangle\}$. This interpretation is named *RR (receive before receive)* in [12] and *receiver semantics* in [18].
RR & SS: this combines the requirements of **RR** and of **SS**: M_1 is sent before M_2 is sent and also M_1 is received before M_2 is received. The set of traces that respect this model is $\{\langle s1, r1, s2, r2\rangle, \langle s1, s2, r1, r2\rangle\}$: both $s1$ comes before $s2$ ("coming before" according to the senders), and $r1$ comes before $r2$ ("coming before" according to the receivers). This interpretation is named *sender-receiver semantics* in [18].
SR: M_1 is sent before M_2 is received. The set of traces that respect this model is $\{\langle s1, r1, s2, r2\rangle, \langle s1, s2, r1, r2\rangle, \langle s1, s2, r2, r1\rangle, \langle s2, s1, r1, r2\rangle, \langle s2, s1, r2, r1\rangle\}$. This interpretation is named *SR (send before receive)* in [12]

It is easy to see that the following inclusions among asynchronous models hold: **RS** $\subset$ **RR & SS** $\subset$ **SS** $\subset$ **SR** and **RS** $\subset$ **RR & SS** $\subset$ **RR** $\subset$ **SR**. The **SS** and **RR** interpretations are not comparable. In the remainder of this paper we consider only the four interpretations defined by Desai & Singh, i.e. we do not consider "RR & SS".

3 Defining Enactability using a Semantic Approach

Basic Notation. In the following, let *ComModel* $= \{CM1, CM2, CM3, CM4, CM5, CM6\}$ be the set of possible (asynchronous) communication models, and *MOISet* $= \{$*SS*, *SR*, *RS*, *RR* $\}$ the set of possible message order interpretations that can be imposed. We also define $\mathcal{A} = \{a, b, c, d, a_1, a_2, \ldots, a_n\}$ to be the set of agents involved in the interaction protocol.

Recall that we consider both interaction and message protocols. When we say that τ is an *interaction* protocol, we mean that the protocol represents sequences of *interaction events*. The set of traces recognized is obtained following the semantics defined in Section 2, and for an interaction protocol τ we define[6] $\mathcal{I}(\tau)$ to be the set of interactions involved in the interaction protocol: $\mathcal{I}(\tau) = \{i \mid \exists I : I \in [\![\tau]\!] \wedge i \in I\}$.

[6] We use "$\in$" to also denote membership of an item in a sequence.

We define $\mathcal{I}$ to be the set of all possible interaction events. Similarly, when τ is a *message* protocol, it represents sequences of send and receive events of the form $aM!$ (send event) and $bM?$ (receive event), and given a particular set of possible interactions $\mathcal{I}$, we define $\mathcal{E}_\mathcal{I}$ to be the corresponding set of events: $\mathcal{E}_\mathcal{I} = \{aM! \mid \exists_{b \in \mathcal{A}} \,.\, a \stackrel{M}{\Longrightarrow} b \in \mathcal{I}\} \cup \{bM? \mid \exists_{a \in \mathcal{A}} \,.\, a \stackrel{M}{\Longrightarrow} b \in \mathcal{I}\}$. In a message protocol τ we have that $E \in [\![\tau]\!] \implies \forall_{e \in E} \,.\, e \in \mathcal{E}_{\mathcal{I}(\tau)}$. Given a message protocol τ we also define $\mathcal{E}(\tau)$ to be the set of message events that occur in the protocol.

Next, we define the language of traces (i.e. of *sequences* of events) for interaction protocols and message protocols. For interaction protocols, the set of all possible traces is defined to be[7]: $\mathcal{L}_\mathcal{I} = \mathcal{I}^* \cup \mathcal{I}^\omega$. For message protocols the definition is somewhat more complex, since there is a relationship between a send and a receive event. Specifically, the set of all possible traces of events is constrained so that a message being received must be preceded by that message having been sent. We also constrain the set so that each message can be sent at most once, and received at most once (i.e. message names are unique). The assumption is made by most authors, see [10] for example, and is considered harmless, since we can integrate many elements to the notion of "message name", such as content, protocol ID and conversation ID, to discriminate between messages at design time. Formally (where $dom(E)$ is standard notation for the domain of a function, here viewing a sequence as a function from numbers to elements):

$$\begin{aligned}\mathcal{L}_{\mathcal{E}_\mathcal{I}} = \{E \in \mathcal{E}_\mathcal{I}^* \cup \mathcal{E}_\mathcal{I}^\omega \mid \\ (\forall_{i,j \in dom(E)} \,.\, E[i] = aM! \wedge E[j] = aM! \implies i = j) \wedge \\ (\forall_{i,j \in dom(E)} \,.\, E[i] = bM? \wedge E[j] = bM? \implies i = j) \wedge \\ (\forall_{i \in dom(E)} \,.\, E[i] = bM? \implies (\exists_{j \in dom(E)} \,.\, E[j] = aM! \wedge j < i))\end{aligned}$$

Message Order Interpretation (MOI). As discussed earlier, we follow prior work in considering four message ordering interpretations (*SS*, *SR*, *RS*, and *RR*). We formalise this by defining a variant semantics that takes an *interaction* protocol τ and returns its semantics in terms of *events* rather than interactions. The possible sequences of events are constrained: given a situation where τ specifies that M_1 must occur before M_2, we constrain the possible sequence of events with the appropriate constraint on events corresponding to the selected MOI.

Definition 1 (Order on interactions in a trace). *Let $I \in \mathcal{L}_\mathcal{I}$ be a trace of interaction events, $E \in \mathcal{L}_{\mathcal{E}_\mathcal{I}}$ be a trace of send and receive events,* moi $\in$ MOISet *a message ordering interpretation, and $a \stackrel{M_1}{\Longrightarrow} b \in \mathcal{I}$, $c \stackrel{M_2}{\Longrightarrow} d \in \mathcal{I}$ two interactions. Abbreviating $a \stackrel{M_1}{\Longrightarrow} b$ as I_1 and $c \stackrel{M_2}{\Longrightarrow} d$ as I_2, we define the message ordering interpretation constraint, denoted $I_1 \prec^E_{\text{moi}} I_2$, as follows:*

$$I_1 \prec^E_{\text{SS}} I_2 \text{ iff } aM_1! \prec_E cM_2! \qquad I_1 \prec^E_{\text{SR}} I_2 \text{ iff } aM_1! \prec_E dM_2?$$
$$I_1 \prec^E_{\text{RS}} I_2 \text{ iff } bM_1? \prec_E cM_2! \qquad I_1 \prec^E_{\text{RR}} I_2 \text{ iff } bM_1? \prec_E dM_2?$$

where $e_1 \prec_E e_2$ iff $\exists_{i,j \in dom(E)} \,.\, E[i] = e_1 \wedge E[j] = e_2 \wedge i < j$ is the constraint that in event trace E the event e_1 occurs before e_2.

[7] The superscripts $*$ and ω are standard notations for (respectively) all finite (all infinite) sequences built from a given set.

Formalising the MOI is not as simple as it might seem. An obvious approach that does not work is to compute the semantics of the interaction protocol τ, and then map each sequence $I \in [\![\tau]\!]$ to a set of message event traces. This does not work because the trace is linear, and therefore a total order, whereas a protocol can specify a partial order[8] (and indeed, in the case of the SR MOI, the ordering may not even be partial, since SR is not transitive). Instead, we define a variant semantics, denoted $[\![\tau]\!]_{moi}$, which is compositional. The semantics follow the standard semantics (Section 2) with a few exceptions. Firstly, the semantics of an interaction I is given as the sequence of sending the message, followed by receiving it (denoted $s(I)$ and $r(I)$, respectively). Secondly, the semantics for a sequence $\tau_1 \cdot \tau_2$ is given in terms of the semantics of τ_1 and τ_2. These are then combined by interleaving them (rather than simply concatenating them), but with the constraint that the result must satisfy the appropriate MOI constraint ($I_1 \prec^E_{moi} I_2$) for all possible final messages of τ_1 (I_1) and all possible initial messages of τ_2 (I_2). Determining initial and final messages is itself somewhat complex, and is done using partially ordered sets.

A partially ordered set (poset) is a pair $(E, <)$ where E is the set of elements (in this case interactions) and $<$ is a transitive binary relation on E. We define the union operator to act piecewise on posets, and to take the transitive closure of the resulting relation, i.e. $(E_1, <_1) \cup (E_2, <_2) = (E_1 \cup E_2, (<_1 \cup <_2)^*)$. The sets of minimal and maximal elements of a poset P are denoted $\min(P)$ and $\max(P)$, respectively.

We can then define the poset of an interaction protocol as follows:

$$\begin{aligned}
\mathrm{poset}(\epsilon) &= (\varnothing, \varnothing) \\
\mathrm{poset}(I) &= (\{I\}, \varnothing) \\
\mathrm{poset}(\tau_1 \wedge \tau_2) &= \mathrm{poset}(\tau_1) \cup \mathrm{poset}(\tau_2) \\
\mathrm{poset}(\tau_1 \mid \tau_2) &= \mathrm{poset}(\tau_1) \cup \mathrm{poset}(\tau_2) \\
\mathrm{poset}(\tau_1 \vee \tau_2) &= \mathrm{poset}(\tau_1) \cup \mathrm{poset}(\tau_2) \\
\mathrm{poset}(\tau_1 \cdot \tau_2) &= \mathrm{poset}(\tau_1) \cdot \mathrm{poset}(\tau_2) \\
(E_1, <_1) \cdot (E_2, <_2) &= (E_1 \cup E_2, <_1 \cup <_2 \cup E_1 \times E_2)
\end{aligned}$$

where we define a sequence of two posets $(E_1, <_1) \cdot (E_2, <_2)$ by collecting the orderings of each of E_1 and E_2, and adding additional ordering constraints between every element of E_1 and every element of E_2. We can now proceed to define the variant compositional semantics $[\![\tau]\!]_{moi}$.

[8] An illustrative example is $\tau = (M_1 \cdot M_2) \mid M_3$. This simple protocol has three sequences of interactions: $\{\langle M_1, M_2, M_3\rangle, \langle M_1, M_3, M_2\rangle, \langle M_3, M_1, M_2\rangle\}$. Assuming an RS message ordering interpretation, then each of the message sequences corresponds to exactly one sequence of events, giving (where we abbreviate sending and receiving M as respectively $M!$ and $M?$): $\{\langle M_1!, M_1?, M_2!, M_2?, M_3!, M_3?\rangle$, $\langle M_1!, M_1?, M_3!, M_3?, M_2!, M_2?\rangle, \langle M_3!, M_3?, M_1!, M_1?, M_2!, M_2?\rangle\}$. However, the protocol does not specify any constraint on M_3, so should also allow other interpretations where the occurrences of $M_3!$ and $M_3!$ are not constrained relative to the other events, for example $\langle M_1!, M_1!, M_3!, M_2!, M_2?, M_3?\rangle$.

$$
\begin{aligned}
[\![\epsilon]\!]_{moi} &= \{\epsilon\} \\
[\![I]\!]_{moi} &= \{\langle s(I), r(I)\rangle\} \\
[\![\tau_1 \vee \tau_2]\!]_{moi} &= [\![\tau_1]\!]_{moi} \cup [\![\tau_1]\!]_{moi} \\
[\![\tau_1 \wedge \tau_2]\!]_{moi} &= [\![\tau_1]\!]_{moi} \cap [\![\tau_1]\!]_{moi} \\
[\![\tau_1 \cdot \tau_2]\!]_{moi} &= \{t \mid t_1 \in [\![\tau_1]\!]_{moi} \wedge t_2 \in [\![\tau_2]\!]_{moi} \wedge t \in t_1 \bowtie t_2 \wedge \\
&\qquad \forall I_1 \in \max(\mathrm{poset}(\tau_1)), \forall I_2 \in \min(\mathrm{poset}(\tau_2)) \,.\, I_1 \prec^{t}_{moi} I_2\} \\
[\![\tau_1 | \tau_2]\!]_{moi} &= \{z \mid t_1 \in [\![\tau_1]\!]_{moi} \wedge t_2 \in [\![\tau_2]\!]_{moi} \wedge z \in t_1 \bowtie t_2\}
\end{aligned}
$$

Where $t_1 \bowtie t_2$ is the set of all interleavings of t_1 and t_2.

Communication Model Semantics. We formalise the defined communication model semantics by defining for each communication model CMi a corresponding language of event traces that incorporates the appropriate restriction, ruling out event sequences that violate the communication model. For example, for $CM1$ the constraint is that immediately after each sending event in u we have its corresponding receiving event, with nothing in the middle; etc. Note that each $\mathcal{L}^{\mathcal{E}_\mathcal{I}}_{CMi}$ takes as a parameter the set of message events $\mathcal{E}_\mathcal{I}$.

$$
\begin{aligned}
\mathcal{L}^{\mathcal{E}_\mathcal{I}}_{CM1} &= \{E \in \mathcal{L}_{\mathcal{E}_\mathcal{I}} | \forall_{a \overset{M_1}{\Longrightarrow} b \in \mathcal{I}} . \forall_{k \in dom(E)} \,.\, aM_1! = E[k-1] \implies bM_1? = E[k]\} \\
\mathcal{L}^{\mathcal{E}_\mathcal{I}}_{CM2} &= \{E \in \mathcal{L}_{\mathcal{E}_\mathcal{I}} | \forall_{a \overset{M_1}{\Longrightarrow} b \in \mathcal{I}} . \forall_{c \overset{M_2}{\Longrightarrow} d \in \mathcal{I}} . \forall_{i,j,k,l \in dom(E)} \,.\, (bM_1? = E[i] \wedge \\
&\qquad dM_2? = E[j] \wedge aM_1! = E[k] \wedge cM_2! = E[l] \wedge k < l) \implies i < j\} \\
\mathcal{L}^{\mathcal{E}_\mathcal{I}}_{CM3} &= \{E \in \mathcal{L}_{\mathcal{E}_\mathcal{I}} | \forall_{a \overset{M_1}{\Longrightarrow} b \in \mathcal{I}} . \forall_{a \overset{M_2}{\Longrightarrow} d \in \mathcal{I}} . \forall_{i,j,k,l \in dom(E)} \,.\, (bM_1? = E[i] \wedge \\
&\qquad dM_2? = E[j] \wedge aM_1! = E[k] \wedge aM_2! = E[l] \wedge k < l) \implies i < j\} \\
\mathcal{L}^{\mathcal{E}_\mathcal{I}}_{CM4} &= \{E \in \mathcal{L}_{\mathcal{E}_\mathcal{I}} | \forall_{a \overset{M_1}{\Longrightarrow} b \in \mathcal{I}} . \forall_{c \overset{M_2}{\Longrightarrow} b \in \mathcal{I}} . \forall_{i,j,k,l \in dom(E)} \,.\, (bM_1? = E[i] \wedge \\
&\qquad bM_2? = E[j] \wedge aM_1! = E[k] \wedge cM_2! = E[l] \wedge k < l) \implies i < j\} \\
\mathcal{L}^{\mathcal{E}_\mathcal{I}}_{CM5} &= \{E \in \mathcal{L}_{\mathcal{E}_\mathcal{I}} | \forall_{a \overset{M_1}{\Longrightarrow} b \in \mathcal{I}} . \forall_{a \overset{M_2}{\Longrightarrow} b \in \mathcal{I}} . \forall_{i,j,k,l \in dom(E)} \,.\, (bM_1? = E[i] \wedge \\
&\qquad bM_2? = E[j] \wedge aM_1! \prec^{E}_{Causal} aM_2!) \implies i < j\} \\
&\qquad \text{where } aM_1! \prec^{u}_{Causal} bM_2! \iff \\
&\qquad\qquad ((a = b \vee M_1 = M_2) \wedge \\
&\qquad\qquad \exists_{i,j \in dom(u)} . (u[i] = aM_1! \wedge bM_2! = u[j] \wedge i < j)) \\
&\qquad\qquad \vee \ (\exists_{ev \in E} . aM_1! \prec^{u}_{Causal} ev \wedge ev \prec^{u}_{Causal} bM_2!) \\
\mathcal{L}^{\mathcal{E}_\mathcal{I}}_{CM6} &= \mathcal{L}_{\mathcal{E}_\mathcal{I}}
\end{aligned}
$$

We can then apply a particular communication model to an *interaction* protocol τ_i using $[\![\tau_i]\!]^{CM}_{moi}$, and to a *message* protocol τ_m using $[\![\tau_m]\!]^{CM}$, which are defined as follows[9].

[9] Note that in the first line we have an *interaction* protocol τ_i, and so the set of message events is given by determining the set of interaction events $\mathcal{I}(\tau)$, and then determining the set of

$$[\![\tau_i]\!]^{CM}_{moi} = [\![\tau_i]\!]_{moi} \cap \mathcal{L}^{\mathcal{E}_{\mathcal{I}(\tau)}}_{CM}$$
$$[\![\tau_m]\!]^{CM} = [\![\tau_m]\!] \cap \mathcal{L}^{\mathcal{E}(\tau)}_{CM}$$

Projection. Projection is defined, intuitively, as focusing on the aspects of the protocol that are relevant for a given role. It is defined as follows, where we write τ^A to denote projecting trace τ for role A.

$$(\epsilon)^A = \epsilon$$
$$(a \overset{M}{\Longrightarrow} b)^A = \begin{cases} aM!, \text{if } a = A \\ bM?, \text{if } b = A \\ \epsilon, \text{otherwise} \end{cases}$$
$$(aM!)^A = \begin{cases} aM!, \text{ if } a = A \\ \epsilon, \text{otherwise} \end{cases}$$
$$(aM?)^A = \begin{cases} aM?, \text{ if } a = A \\ \epsilon, \text{otherwise} \end{cases}$$
$$(\tau_1 \otimes \tau_2)^A = (\tau_1)^A \otimes (\tau_2)^A \text{ (where } \otimes \text{ is any operator)}$$

We then define the *distribution* of τ, denoted $\ulcorner \tau \urcorner$, where τ involves roles $a_1 \dots a_n$ as[10]:

$$\ulcorner \tau \urcorner = \tau^{a_1} \| \cdots \| \tau^{a_n}$$

To give an example, let us consider again the scenario proposed in Section 1. Alice decided to book Carol's apartment and now Carol needs some information from Alice in order to complete the reservation. This information can be wrong or incomplete, in which case Carol gives Alice an opportunity to amend the information, and in either case the interaction then concludes with Carol confirming the booking. This can be represented as the following specification:

$$\begin{aligned} \mathit{reqInfo} = \mathit{Alice} \overset{Info}{\Longrightarrow} \mathit{Carol} \ \cdot \\ (\mathit{Carol} \overset{Wrong}{\Longrightarrow} \mathit{Alice} \ \cdot \ \mathit{Alice} \overset{Info'}{\Longrightarrow} \mathit{Carol} \ \vee \ \epsilon) \ \cdot \\ \mathit{Carol} \overset{Booked}{\Longrightarrow} \mathit{Alice} \end{aligned}$$

Let us consider $main$ as the sequential combination of the two protocols: $main = modifyRes \ \cdot \ reqInfo$. Then the projection of $main$ on each single agent gives the following distribution.

---

message events $\mathcal{E}_{\mathcal{I}(\tau)}$. By contrast, in the second line, τ_m is a *message* protocol, so we just determine the set of message events directly ($\mathcal{E}(\tau)$).

[10] We use $\|$ to distinguish between parallel composition of different agents, and parallel composition within a protocol. This distinction is used later in this section.

$$
\begin{aligned}
\ulcorner main \urcorner &= main^{Alice} \parallel main^{Bob} \parallel main^{Carol} \\
main^{Alice} &= modifyRes^{Alice} \cdot reqInfo^{Alice} \\
modifyRes^{Alice} &= AliceCanc! \cdot AliceRes! \\
reqInfo^{Alice} &= AliceInfo! \cdot (AliceWrong? \cdot AliceInfo'! \vee \epsilon) \cdot \\
&\quad AliceBooked? \\
main^{Bob} &= modifyRes^{Bob} \cdot reqInfo^{Bob} = BobCanc? \cdot \epsilon \\
main^{Carol} &= modifyRes^{Carol} \cdot reqInfo^{Carol} \\
modifyRes^{Carol} &= CarolRes? \\
reqInfo^{Carol} &= CarolInfo? \cdot (CarolWrong! \cdot CarolInfo'? \vee \epsilon) \cdot \\
&\quad CarolBooked!
\end{aligned}
$$

In order to define the semantics of a projected protocol we need to first define what we term a *decision structure*. This is needed in the semantics in order to deal correctly with projected protocols. Specifically, the intuition for enactability (see Section 3) is that an interaction protocol τ involving, say, three roles a, b and c is enactable iff there exist three protocols τ^a, τ^b and τ^c such that their concurrent interleaving results in the same behaviour as the original protocol. However, when a protocol contains choices ($\vee$) we need to ensure that the occurrences of $\vee$ in each of τ^a, τ^b and τ^c arising from the same $\vee$ in τ are treated consistently. For example, consider the protocol $\tau = a \overset{M_1}{\Longrightarrow} b \vee a \overset{M_2}{\Longrightarrow} c$. This protocol is simple: it specifies that agent a can either send a message (M_1) to b, or it can send a different message ("M_2") to agent c. When we distribute the protocol by projecting it (see Section 3) and forming $\tau^a \| \tau^b \| \tau^c$ we obtain the distributed protocol $(aM_1! \vee aM_2!) \| (bM_1? \vee \varepsilon) \| (\varepsilon \vee cM_2?)$. However, if we interpret each $\vee$ independently (as the semantics would naturally do) then we can have *inconsistent* choices. For example, we could have $(aM_1!) \| (\varepsilon) \| (\varepsilon)$ where the message is sent by a, but b does not elect to receive it. So what we need to do is ensure that each of the three occurrences of "$\vee$" represent the *same* choice, and that the choice should be made consistently.

The heart of the issue is that the trace expression notation offers a choice operator ($\vee$), which is adequate for global protocols. However, for local protocols it is important to be able to distinguish between a choice that represents a free (local) choice, and a choice that is forced by earlier choices. In this example, a can freely choose whether to send M_1 or M_2. However, the choice of b whether to receive M_1 or not is not a free choice, but is forced by a's earlier choice.

Our semantics handles this by defining a *decision structure* which is used to enforce consistent choices. Formally, given a protocol τ we define $d(\tau)$ as a set of *decision structures* (formal definition below). A decision structure is a syntactic structure that mirrors the structure of τ, except that each $\vee$ is annotated with a decision (e.g. L or R). We define three operations on a decision structure: to get the sub-decision structure corresponding to the left part (denoted $d.L$), to get the right part ($d.R$) and to get the decision (L or R) associated with the current $\vee$ node (denoted $d.D$). We define $d(\tau)$ to create a set of decision structures, each of which corresponds to the structure of τ,

but where all possible assignments of decisions are made. Observe that If τ contains N occurrences of $\vee$ then the set $d(\tau)$ contains 2^N elements.

For example, given $\tau = a \stackrel{M_1}{\Longrightarrow} b \vee a \stackrel{M_2}{\Longrightarrow} b$ we have that $d(\tau) = \{\_ \overset{L}{\vee} \_, \_ \overset{R}{\vee} \_\}$ where we use $\_$ to indicate an irrelevant part of a decision structure, and $\overset{L}{\vee}$ to denote a node tagged with a decision L.

In addition to decisions of L and R, the definition of $d(\tau_1 \vee \tau_2)$ has a second case ($\ldots \cup \{t_1 \overset{LR}{\vee} t_2 \mid \ldots\}$). The reason is that it is only possible to enforce consistent choice if the choice is made by a single agent. If this is not the case, then we annotate with "LR" to indicate that a mixed choice is possible. For example, given $\tau = b \stackrel{M_1}{\Longrightarrow} a \vee a \stackrel{M_2}{\Longrightarrow} b$ we have that $d(\tau) = \{\_ \overset{LR}{\vee} \_\}$ because the agents associated with the set of possible initial messages in each branch are different ($ag(\tau_1) = \{b\} \neq ag(\tau_2) = \{a\}$).

$$
\begin{aligned}
d(\varepsilon) &= \{\varepsilon\} \\
d(I) &= \{I\} \\
d(\tau_1 \vee \tau_2) &= \{t_1 \overset{x}{\vee} t_2 \,|\, t_1 \in d(\tau_1) \wedge t_2 \in d(\tau_2) \\
&\qquad \wedge x \in \{R, L\} \wedge ag(\tau_1) = ag(\tau_2) \wedge |ag(\tau_1)| = 1\} \\
&\quad \cup \{t_1 \overset{LR}{\vee} t_2 \,|\, t_1 \in d(\tau_1) \wedge t_2 \in d(\tau_2) \\
&\qquad \wedge ((ag(\tau_1) \neq ag(\tau_2)) \vee (|ag(\tau_1)| \neq 1))\} \\
&\quad \text{where } ag(\tau) = \{p \mid p \stackrel{M}{\Longrightarrow} r \in \min(\text{poset}(\tau))\} \\
d(\tau_1 \oplus \tau_2) &= \{t_1 \oplus t_2 \mid t_1 \in d(\tau_1) \wedge t_2 \in d(\tau_2)\} \\
(\tau_L \otimes \tau_R).L = \tau_L \qquad & (\tau_L \otimes \tau_R).R = \tau_R \qquad (\tau_L \overset{X}{\vee} \tau_R).D = X
\end{aligned}
$$

Where $\otimes$ is any operator, and $\oplus$ is any operator other than $\vee$.

We now specify the semantics of a distributed protocol, denoted $[\![\tau]\!]_{\text{dist}}$. The semantics is defined in terms of a union over possible decision structures (first line). The remaining equations for the semantics carry along the decision structure, and follow it in recursive calls, and for the semantics of $\vee$ it enacts the decision specified in the structure, rather than considering both sub-protocols. Note that projection is defined using $\|$ rather than the usual $|$. The difference in the semantics below is that $\|$ passes the *same* decision structure to both arguments. This ensures consistency between agents, but not within agents.

$$
\begin{aligned}
[\![\tau]\!]_{\text{dist}} &= \bigcup_{dt \in d(\tau)} [\![\tau^{a_1} \| \ldots \| \tau^{a_n}]\!]^{dt} \\
[\![M]\!]^{dt} &= \{\langle M \rangle\} \\
[\![\varepsilon]\!]^{dt} &= \{\langle\rangle\} \\
[\![\tau_1 \cdot \tau_2]\!]^{dt} &= \{t_1 \circ t_2 \mid t_1 \in [\![\tau_1]\!]^{dt.L} \wedge t_2 \in [\![\tau_2]\!]^{dt.R}\} \\
[\![\tau_1 \wedge \tau_2]\!]^{dt} &= [\![\tau_1]\!]^{dt.L} \cap [\![\tau_2]\!]^{dt.R} \\
[\![\tau_1 \vee \tau_2]\!]^{dt} &= \text{if } dt.D = R \text{ then } [\![\tau_2]\!]^{dt.R} \text{ else if } dt.D = L \text{ then } [\![\tau_1]\!]^{dt.L} \\
&\qquad \text{else } [\![\tau_2]\!]^{dt.R} \cup [\![\tau_1]\!]^{dt.L}
\end{aligned}
$$

$$[\![\tau_1|\tau_2]\!]^{dt} = \{z \mid t_1 \in [\![\tau_1]\!]^{dt.L} \wedge t_2 \in [\![\tau_2]\!]^{dt.R} \wedge z \in t_1 \bowtie t_2\}$$
$$[\![\tau_1\|\tau_2]\!]^{dt} = \{z \mid t_1 \in [\![\tau_1]\!]^{dt} \wedge t_2 \in [\![\tau_2]\!]^{dt} \wedge z \in t_1 \bowtie t_2\}$$

where $t_1 \bowtie t_2$ is the set of all interleavings of t_1 and t_2, and $\circ$ is concatenation over sequences. Note that if τ does not contain any occurrences of $\vee$ then the semantics above reduce to the standard semantics.

Finally, we define $[\![\tau_i]\!]^{CM}_{\text{dist}}$, which computes the semantics of an interaction protocol τ_i by distributing it, and also applies a particular communication model *CM*.

$$[\![\tau_i]\!]^{CM}_{\text{dist}} = [\![\tau_i]\!]_{\text{dist}} \cap \mathcal{L}_{CM}^{\mathcal{E}_{\mathcal{I}(\tau)}}$$

Enactability. We are now finally in a position to define enactability. The intuition is that an interaction protocol τ is enactable iff the semantics of τ, with respect to a selected message ordering interpretation and communication model, can be realised by a distributed version of the protocol. In other words, if there exists for each role r a corresponding message protocol τ_r such that the combination of these protocols realises the same behaviour as τ. However, instead of considering whether there exists some τ_r, we let $\tau_r = \tau^r$, i.e. we take for each role the projected protocol as its protocol.

We also consider a notion of *weak* enactability. This applies in a situation where the distributed enactment is able to avoid violating the behaviour specified by τ, but is not able to recreate all of the behaviours that τ specifies. In other words, if a protocol is weakly enactable, the interleaving of the corresponding local protocols generates a subset of its traces (with a fixed *moi* and communication model). This means that a distributed implementation of the protocol can be sound (generates only valid traces), but cannot be complete (not all the traces are generated). This situation can arise with weaker message ordering interpretations (see below for examples). Weak enactability can also arise in situations where two ordered messages have two overlapping roles (e.g. $\tau = a \xRightarrow{M_1} b \cdot b \xRightarrow{M_2} a$). In this situation the projection operator is too strict: it has $\tau^b = r(M_1) \cdot s(M_2)$, but if we adopt an SR message ordering interpretation, then we do not need to ensure that M_2 is sent after M_1 is received, only that M_1 is sent before M_2 is received, which role a can ensure on its own.

Definition 2 (Strongly/Weakly Enactable). *Let τ be an interaction protocol, $\{a_1, a_2, ..., a_n\}$ the set of agents involved in τ,* moi $\in$ MOISet *a message order interpretation and* CM $\in$ ComModel *a communication model. We say that, τ is strongly (weakly) enactable, for* moi *semantics in* CM *model iff the decomposition of τ through projection on its agents $\{a_1, a_2, ..., a_n\}$ recognizes the same (a subset of) traces recognized by τ. Formally:*

$$enact(\tau)^{\text{CM}}_{\text{moi}} \ \textit{iff} \ [\![\tau]\!]^{\text{CM}}_{\text{dist}} = [\![\tau]\!]^{\text{CM}}_{\text{moi}}$$
$$weak\_enact(\tau)^{\text{CM}}_{\text{moi}} \ \textit{iff} \ [\![\tau]\!]^{\text{CM}}_{\text{dist}} \subseteq [\![\tau]\!]^{\text{CM}}_{\text{moi}}$$

Figure 1 show the results of applying this definition to a number of cases, with different message ordering interpretation, and different communication models. These tables were all generated by the Haskell implementation of the definitions in this paper, in which ✔and (✔) denote *strongly* and *weakly* enactable, respectively. The prototype

$a \overset{M_1}{\Longrightarrow} b \cdot b \overset{M_5}{\Longrightarrow} c$

CM	RS	RR	SS	SR
CM1	✔	✔	✔	✔
CM2	✔	(✔)	(✔)	(✔)
CM3	✔	(✔)	(✔)	(✔)
CM4	✔	(✔)	(✔)	(✔)
CM5	✔	(✔)	(✔)	(✔)
CM6	✔	(✔)	(✔)	(✔)

$a \overset{M_1}{\Longrightarrow} b \cdot a \overset{M_2}{\Longrightarrow} c$

CM	RS	RR	SS	SR
CM1	✔	✔	✔	✔
CM2	✘	✔	✔	(✔)
CM3	✘	✔	✔	(✔)
CM4	✘	✘	✔	(✔)
CM5	✘	✘	✔	(✔)
CM6	✘	✘	✔	(✔)

$a \overset{M_1}{\Longrightarrow} b \cdot c \overset{M_6}{\Longrightarrow} b$

CM	RS	RR	SS	SR
CM1	✔	✔	✔	✔
CM2	✘	✔	✔	(✔)
CM3	✘	✔	✘	(✔)
CM4	✘	✔	✔	(✔)
CM5	✘	✔	✘	(✔)
CM6	✘	✔	✘	(✔)

$a \overset{M_1}{\Longrightarrow} b \cdot c \overset{M_4}{\Longrightarrow} a$

CM	RS	RR	SS	SR
CM1	✔	✔	✔	✔
CM2	✘	✘	✘	✔
CM3	✘	✘	✘	✔
CM4	✘	✘	✘	✔
CM5	✘	✘	✘	✔
CM6	✘	✘	✘	✔

$a \overset{M_1}{\Longrightarrow} b \cdot a \overset{M_2}{\Longrightarrow} b$

CM	RS	RR	SS	SR
CM1	✔	✔	✔	✔
CM2	✘	✔	✔	(✔)
CM3	✘	✔	✔	(✔)
CM4	✘	✔	✔	(✔)
CM5	✘	✔	✔	(✔)
CM6	✘	(✔)	(✔)	(✔)

$a \overset{M_1}{\Longrightarrow} b \cdot b \overset{M_3}{\Longrightarrow} a$

CM	RS	RR	SS	SR
CM1	✔	✔	✔	✔
CM2	✔	(✔)	(✔)	(✔)
CM3	✔	(✔)	(✔)	(✔)
CM4	✔	(✔)	(✔)	(✔)
CM5	✔	(✔)	(✔)	(✔)
CM6	✔	(✔)	(✔)	(✔)

$a \overset{M_1}{\Longrightarrow} b \vee a \overset{M_2}{\Longrightarrow} c$

CM	RS	RR	SS	SR
CM1	✔	✔	✔	✔
CM2	✔	✔	✔	✔
CM3	✔	✔	✔	✔
CM4	✔	✔	✔	✔
CM5	✔	✔	✔	✔
CM6	✔	✔	✔	✔

$a \overset{M_1}{\Longrightarrow} b \vee b \overset{M_3}{\Longrightarrow} a$

CM	RS	RR	SS	SR
CM1	✔	✔	✔	✔
CM2	✘	✘	✘	✘
CM3	✘	✘	✘	✘
CM4	✘	✘	✘	✘
CM5	✘	✘	✘	✘
CM6	✘	✘	✘	✘

Fig. 1. Automatically generated analyses of enactability

counts ˜300 LOC. It implements the trace expression standard semantics, message order interpretation, communication model semantics and enactability check[11].

Looking at the tables in Figure 1, we make the following observations.

Firstly, CM1 is quite strict: all the cases considered are enactable under CM1, regardless of the selected message ordering interpretation. This is expected: we know that CM1 is quite strong.

Secondly, for many examples there is not a difference in enactability with the different communication models (other than CM1). The exception is where the communication model corresponds to the combination of MOI and the pattern in the protocol. For example, in the top row, second table from the right, the simple protocol is enactable given the SS message ordering interpretation only with CM2 and CM4 (and, of course, CM1). This is because, for this protocol, both messages are received by the same agent but sent by different agents, and, given an RR MOI, the desired constraint that agent B receives the first message before the second, can only be enforced using a communication model that guarantees delivery of messages to the same recipient in the order in which messages were sent. Both CM2 and CM4 provide this guarantee (in fact CM4 provides exactly this, and CM2 is stronger).

Thirdly, RS appears to be a good choice for message ordering interpretation, since it is the only MOI where protocols are never weakly enactable. For the other message ordering interpretations, there are protocols that are only weakly enactable (for communication models other than CM1). A protocol being weakly enactable indicates that the desired behaviour specified by the MOI is too loose: it permits behaviours that the distributed realisation cannot realise. On the other hand, in the case of the left-most table on the bottom row (protocol $a \overset{M_1}{\Longrightarrow} b \cdot a \overset{M_2}{\Longrightarrow} b$), the protocol is not enactable under RS (except for CM1), but is enactable under SS and under RR. Turning to SR, we observe that it seems to be too weak: almost all the protocols in the figure are enactable (although in most cases only weakly enactable).

[11] The code is available (anonymously) on the web at: `http://enactability.altervista.org/`

Returning to the example from the introduction:

$$modifyRes = Alice \overset{Canc}{\Longrightarrow} Bob \cdot Alice \overset{Res}{\Longrightarrow} Carol$$

this corresponds to the second table from the left in the top row of Figure 1. This shows that, if one desires an RR MOI, then the underlying message communication must be $CM1$, $CM2$ or $CM3$, in order for the protocol to be enactable.

4 Discussion

Despite the large amount of work on enactability, very few approaches consider how message ordering and decision structures affect its definition, very few come with an implemented prototype, and none considers the issues raised by the communication model.

Taking all these features into account in a unified semantic-driven way, and demonstrating the potential of the approach on a highly expressive protocol language, are the innovative and original features of this contribution.

Desai and Singh [12] limit their investigation to the RS message ordering interpretation, which they consider the standard of correctness. Hence, despite the introduction they provide to other message orderings and to the problems they might raise, the definition of enactability they provide is not parametric in the MOI.

Lanese *et al.* [18] move a step further, but the generality of their approach is still limited. They define three different notions of enactability, which they name conformance: sender conformance, receiver conformance, and disjoint conformance. That approach is more flexible than the one by Desai and Singh, but less general than ours, where the definition of enactability is parametric in the MOI and does not require different cases. Also, they only consider how sequence and choice are affected by MOIs, leaving the study of other operators for the future. Moreover, when discussing interaction protocols whose most external operator is a choice, they put a very strong constraint for enactability, namely that the agents involved in the two branches of the choice (excluding the agents involved in the choice itself) are the same. We added decision structures to overcome this restriction, and provide a notion of enactability that can succeed even when that constraint is not met.

Neither Desai and Singh, nor Lanese *et al.*, use formalisms for protocol representation as expressive as trace expressions, and neither of them present experiments obtained from a working prototype, as we do.

With respect to the introduction of decision structures to remove unnecessary restrictions on enactability of protocols when choice is involved, our proposal is similar to that by Qiu *et al.*, [21]. However, as for the other works we have discussed in this section, we implemented our enactability checker, whereas their work only provides definitions. Additionally, our approach is simpler in that we do not need to label the choice operator with agents as they do, and, finally, they do not consider as general a setting (with a range of message ordering interpretations and communication semantics).

In the future, we will address both theoretical and practical issues. On the theoretical side, we will carry out a systematic analysis of the relationships between the communication model and the message ordering interpretation, to identify those combinations that provide some guarantees by design. We will also explore the relationship between enactability and distributed monitorability [14], since the two notions are related.

On the practical side, we plan to improve our working prototype to provide a tool to assess protocols for enactability. Apart from providing a user-friendly interface, a key issue to address will be to provide a way to isolate the part of a non-enactable protocol that makes it non-enactable. Also, trace expressions are interpreted in a coinductive way [23] to represent infinite traces of events. Since Haskell does not support coinduction, the existing prototype can be only used on acyclic message and interaction protocols. Haskell has been chosen because the implementation mimics the semantics, which makes it easy to check that the Haskell implementation correctly implements the formal definitions. In order to fully implement the proposed features we are planning to develop the enactability check using SWI-Prolog[12], which natively supports coinduction. We also will explore alternative approaches to dealing with cyclic trace expressions, including the possibility of translating them to (e.g.) Büchi automata. Additionally, to stress-test the prototype and assess its performance from a qualitative and quantitative viewpoint we plan to create a library of interaction protocols known to be "problematic" with respect to enactability, and perform systematic experiments.

Finally, this work highlighted the need to characterise existing agent infrastructures such as JADE [5], Jason [6] and Jadex [20] in terms of the communication models they support. We asked the developers of the three frameworks, and all agreed that they support the CM4 model, which was the answer we expected. Nevertheless, this answer was far from being trivial to identify for the developers themselves. As an example, Lars Braubach pointed out that Jadex uses service interaction on top of messages, i.e. communication is fully asynchronous but based on interfaces and method calls from a user perspective, which makes answering the question more subtle than it might seem. Both Jomi Fred Hübner (Jason) and Agostino Poggi (JADE) recognized that they had to spend some time on the issue, also because the classification CM0-CM6 based on [10] requires time to be read and understood. This suggests two further directions of work. On the one hand, we might run experiments on the three platforms above, to confirm their CM and try to check if other models are (unexpectedly) supported. On the other, the EMAS community might devise a standard taxonomy for CMs, such as the as one in [10], and provide each platform with a set of agreed upon "platform standard metadata" (how many agents can run concurrently without experiencing problems; learning curve for different types of professionals; known practical applications; etc). These metadata should include CM as well. This piece of information, along with the approach we have proposed in this paper, would allow the developers to determine whether a protocol is enactable on a given infrastructure.

[12] `http://www.swi-prolog.org`

Acknowledgements

We thank Lars Braubach, Jomi Fred Hübner, and Agostino Poggi for their support in understanding the communication model supported by Jadex, Jason, and JADE.

References

1. Ancona, D., Drossopoulou, S., Mascardi, V.: Automatic generation of self-monitoring MASs from multiparty global session types in Jason. In: DALT. LNCS, vol. 7784, pp. 76–95. Springer (2012)
2. Ancona, D., Ferrando, A., Franceschini, L., Mascardi, V.: Parametric trace expressions for runtime verification of Java-like programs. In: FTfJP@ECOOP. pp. 10:1–10:6. ACM (2017)
3. Ancona, D., Ferrando, A., Mascardi, V.: Comparing trace expressions and linear temporal logic for runtime verification. In: Theory and Practice of Formal Methods. LNCS, vol. 9660, pp. 47–64 (2016)
4. Autili, M., Tivoli, M.: Distributed enforcement of service choreographies. In: Cámara, J., Proença, J. (eds.) 13th International Workshop on Foundations of Coordination Languages and Self-Adaptive Systems (FOCLASA). EPTCS, vol. 175, pp. 18–35 (2014). doi:http://doi.org/10.4204/EPTCS.175.210.4204/EPTCS.175.2
5. Bellifemine, F.L., Caire, G., Greenwood, D.: Developing Multi-Agent Systems with JADE. Wiley (2007)
6. Bordini, R.H., Hübner, J.F., Wooldridge, M.: Programming Multi-Agent Systems in AgentSpeak Using Jason (Wiley Series in Agent Technology). John Wiley & Sons (2007)
7. Broda, S., Machiavelo, A., Moreira, N., Reis, R.: Automata for regular expressions with shuffle. Information and Computation **259**(2), 162–173 (2018)
8. Carbone, M., Honda, K., Yoshida, N.: Structured communication-centred programming for web services. In: Nicola, R.D. (ed.) 16th European Symposium on Programming (ESOP). LNCS, vol. 4421, pp. 2–17. Springer (2007). doi:http://doi.org/10.1007/978-3-540-71316-6_210.1007/978-3-540-71316-6_2
9. Castagna, G., Dezani-Ciancaglini, M., Padovani, L.: On global types and multi-party sessions. In: Bruni, R., Dingel, J. (eds.) FMOODS/FORTE. LNCS, vol. 6722, pp. 1–28. Springer (2011). doi:http://doi.org/10.1007/978-3-642-21461-5_110.1007/978-3-642-21461-5_1
10. Chevrou, F., Hurault, A., Quéinnec, P.: On the diversity of asynchronous communication. Formal Aspects of Computing **28**(5), 847–879 (2016). doi:http://doi.org/10.1007/s00165-016-0379-x10.1007/s00165-016-0379-x
11. Decker, G., Weske, M.: Local enforceability in interaction Petri nets. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) 5th International Conference on Business Process Management. LNCS, vol. 4714, pp. 305–319. Springer (2007). doi:http://doi.org/10.1007/978-3-540-75183-0_2210.1007/978-3-540-75183-0_22
12. Desai, N., Singh, M.P.: On the enactability of business protocols. In: Fox, D., Gomes, C.P. (eds.) Twenty-Third AAAI Conference on Artificial Intelligence. pp. 1126–1131. AAAI Press (2008), `http://www.aaai.org/Library/AAAI/2008/aaai08-178.php`
13. Ferrando, A., Ancona, D., Mascardi, V.: Monitoring patients with hypoglycemia using self-adaptive protocol-driven agents: A case study. In: Baldoni, M., Müller, J.P., Nunes, I., Zalila-Wenkstern, R. (eds.) 4th International Workshop on Engineering Multi-Agent Systems (EMAS). LNCS, vol. 10093, pp. 39–58. Springer (2016). doi:http://doi.org/10.1007/978-3-319-50983-9_310.1007/978-3-319-50983-9_3

14. Ferrando, A., Ancona, D., Mascardi, V.: Decentralizing MAS monitoring with DecAMon. In: Larson, K., Winikoff, M., Das, S., Durfee, E.H. (eds.) Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017. pp. 239–248. ACM (2017), http://dl.acm.org/citation.cfm?id=3091164
15. Honda, K., Vasconcelos, V.T., Kubo, M.: Language primitives and type discipline for structured communication-based programming. In: Hankin, C. (ed.) 7th European Symposium on Programming (ESOP). LNCS, vol. 1381, pp. 122–138. Springer (1998). doi:http://doi.org/10.1007/BFb005356710.1007/BFb0053567
16. Honda, K., Yoshida, N., Carbone, M.: Multiparty asynchronous session types. In: Necula, G.C., Wadler, P. (eds.) 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL). pp. 273–284. ACM (2008). doi:http://doi.org/10.1145/1328438.132847210.1145/1328438.1328472
17. Lamport, L.: Time, clocks, and the ordering of events in a distributed system. Communications of the ACM **21**(7), 558–565 (Jul 1978). doi:http://doi.org/10.1145/359545.35956310.1145/359545.359563
18. Lanese, I., Guidi, C., Montesi, F., Zavattaro, G.: Bridging the gap between interaction- and process-oriented choreographies. In: Cerone, A., Gruner, S. (eds.) Sixth IEEE International Conference on Software Engineering and Formal Methods (SEFM). pp. 323–332. IEEE Computer Society (2008). doi:http://doi.org/10.1109/SEFM.2008.1110.1109/SEFM.2008.11
19. Poizat, P., Salaün, G.: Checking the realizability of BPMN 2.0 choreographies. In: 27th Annual ACM Symposium on Applied Computing (SAC). pp. 1927–1934. ACM (2012). doi:http://doi.org/10.1145/2245276.223209510.1145/2245276.2232095
20. Pokahr, A., Braubach, L., Lamersdorf, W.: Jadex: A BDI reasoning engine. In: Bordini, R.H., Dastani, M., Dix, J., El Fallah Seghrouchni, A. (eds.) Multi-Agent Programming: Languages, Platforms and Applications, pp. 149–174. Springer (2005). doi:http://doi.org/10.1007/0-387-26350-0_610.1007/0-387-26350-0_6
21. Qiu, Z., Zhao, X., Cai, C., Yang, H.: Towards the theoretical foundation of choreography. In: Williamson, C.L., Zurko, M.E., Patel-Schneider, P.F., Shenoy, P.J. (eds.) 16th International World Wide Web Conference (WWW). pp. 973–982. ACM (2007). doi:http://doi.org/10.1145/1242572.124270410.1145/1242572.1242704
22. Salaün, G., Bultan, T., Roohi, N.: Realizability of choreographies using process algebra encodings. IEEE Transactions on Services Computing **5**(3), 290–304 (2012). doi:http://doi.org/10.1109/TSC.2011.910.1109/TSC.2011.9
23. Sangiorgi, D.: On the origins of bisimulation and coinduction. ACM Transactions on Programming Languages and Systems **31**(4), 15:1–15:41 (May 2009). doi:http://doi.org/10.1145/1516507.151651010.1145/1516507.1516510
24. Takeuchi, K., Honda, K., Kubo, M.: An interaction-based language and its typing system. In: Halatsis, C., Maritsas, D.G., Philokyprou, G., Theodoridis, S. (eds.) 6th International Conference on Parallel Architectures and Languages Europe (PARLE). LNCS, vol. 817, pp. 398–413. Springer (1994). doi:http://doi.org/10.1007/3-540-58184-7_11810.1007/3-540-58184-7_118