Local Enforceability and Inconsumable Messages in Choreography Models

Vitaly Kozyura  
SAP Research, Bleichstr. 8  
64283 Darmstadt, Germany  
v.kozyura@sap.com

Andreas Roth  
SAP Research, Bleichstr. 8  
64283 Darmstadt, Germany  
andreas.roth@sap.com

Wei Wei  
SAP Research, Bleichstr. 8  
64283 Darmstadt, Germany  
wei01.wei@sap.com

Abstract—Choreography models describe the communication protocols between services. Every choreography model can be considered either from a global or from a local point of view. The global model specifies a high-level view of the conversation between service components, and can be considered as being interpreted from an observer point of view. The local model is derived from the global model, and specifies the communication-relevant behavior of each component. The connection between global and local models is achieved differently by connecting observing steps with either the send steps or with the receive steps. The consistency of the global model and the local model can be assured by the local enforceability property; any behavior that the local model permits is also possible to observe in the global model.

Another important property of choreography models requires the absence of inconsumable messages, i.e., any messages ready to be received may always be received. In this paper, we study the relations between local enforceability and inconsumable messages in case that the local model is obtained from the global model without modification or further constraints. As a part of the conclusions, we show that if a choreography model is free of inconsumable messages, it is also local enforceable no matter how the global model is connected with the local one.

This is a desired result because the mechanical checking of inconsumable messages is much more efficient than that of local enforceability. In case of finite state systems one can check the absence of inconsumable messages in a linear time in the size of the local model, whereas checking local enforceability has an exponential complexity.

Keywords—Software Modeling, Software Verification, Message Choreography, Trace Simulation, Local Enforceability, Inconsumable Messages

I. INTRODUCTION

Enterprise application software, i.e. software which supports companies in their business processes (like accounting, order management, production, etc.), has undergone massive changes in recent years which led from large monolithic applications to distributed architectures consisting of loosely coupled components. To this end service-oriented architectures (SOA) have been introduced which use asynchronous messaging as a means to enable loose coupling. This shift has imposed a number of challenges, one of which is to manage proper (i.e. consistency preserving) communication among the loosely coupled components. In this context, formal modeling and the static verification of message choreographies has become an important topic in the enterprise application domain [1].

In this paper we elaborate on a theoretical result which makes such verification considerably more efficient. We use a framework for modeling message choreographies developed at SAP Research called Message Choreography Modeling (MCM) which is in detail described in [2]. In [3] we reported on how these models can be transformed into a formal representation, such as Event-B, allowing for a formal analysis, such as test generation or formal verification.

MCM models describe the message exchange between exactly two communicating components both on the level of a component-neutral observer (global model) and in terms of the send/receive events of each of both components (local partner model). Global models and local partner models are required to be identical in structure. This property is in line with business requirements (see [4]) and allows for the improvements of the verification process reported in this paper.

Interesting properties for the verification (and relevant for this paper) are two kinds of properties:

1. Local enforceability: any behavior that the local model permits is also possible to observe in the global model.
2. Absence of inconsumable messages: the intended receiver of a message which is being exchanged is always ready to receive the message.

The main contribution of this paper is to prove that (2) implies (1). Moreover, since there are various ways to connect local and global models with each other [2], i.e. does a transition in the global model correspond to a send or to a receive event in the local model, we show that this holds regardless of this choice. In addition, this result is independent from whether messages can change order while being transferred; in the case that the order is preserved and a send-viewpoint is assumed the absence of inconsumable messages is even equivalent to local enforceability.

This contribution is a highly desired result because absence of inconsumable messages can be more efficiently checked than local enforceability. Considering finite state systems (or finite state under-approximations of infinite state systems) it can be verified in linear time (in the size of a local model) whether an inconsumable message exists. Checking
local enforceability has an exponential complexity. Note that the aim of our paper is to investigate the relations between the two above mentioned properties in the context of the MCM language. However, it is out of scope how these two properties can be verified using either theorem proving or model checking techniques.

The structure of the paper is as follows. In Section II we formally define choreography models. Then we motivate the relevant properties Local Enforceability (Section III) and Inconsumable Messages (Section IV). Section V contains the main result of this paper, namely the inclusion relation of choreographies without inconsumable messages within those satisfying local enforceability. The special case of messages preserving order during transfer is covered in Section VI. Section VII discusses related work and Section VIII concludes the paper with a summary and an outlook on future work.

II. CHOREOGRAPHY MODELS

Formally, a choreography model $M = (G, L)$ is a pair of a global choreography model $G$ and a local partner model $L$.

The global model $G$ is a tuple $(S, s_0, T, I, \Rightarrow)$ where (1) $S$ is a finite set of states; (2) $s_0 \in S$ is the initial state; (3) $T \subseteq S$ is a set of target states; (4) $I$ is a finite set of interactions; and (5) $\Rightarrow \subseteq S \times I \times S$ is the transition relation. Additionally, every interaction $i$ corresponds to a certain type of messages, and is associated with a direction of communication. We can use a function $sender(i)$ to indicate which local partner initiates $i$. Figure 1 shows an example of a global model. Every state is represented as a rounded rectangle. The initial state $Start$ is in dotted lines, and the target states $Start$ and $Ordered$ are connected to a circle. Every envelope represents an interaction, and the small arrow inside indicates the direction of the interaction. Transitions are shown by directed lines connecting two states through interactions. As an example, from the state $Reserved$, the occurrence of an interaction $Order$ results in the new state $Ordered$.

![Figure 1. A global choreography model.](image)

A global model $G$ can be seen as a labelled transition system $LTS(G)$ where the transitions are labelled with interactions. The semantics of a global model can then be given as the set of all traces of $LTS(G)$. A trace is a finite or infinite sequence of interactions $(i_1, \ldots, i_n, \ldots)$, which corresponds to a sequence of states $(s_0, s_1, \ldots, s_n, \ldots)$ such that $(s_{k-1}, i_k, s_k) \in \Rightarrow$ for every $i_k$ in the trace. In our example model, the sequence $(Request, Confirm, Order)$ is a trace. In order to connect the global model to the local model, each global trace can be interpreted as the observation of a sequence of send events or a sequence of receive events in the local model, depending on whether the sender or the receiver point of view is adopted.

The local model $L$ consists of two communication partners and four communication channels available for exchanging messages. Each communication partner has a control structure identical to that of the global model, except that each interaction is replaced with either a send or a receive event depending on who initiates the interaction. Formally, the local model $L$ is a triple $(L_1, L_2, Ch)$ where (1) for each communicating partner $P_k$ ($k \in \{1, 2\}$), $L_k = (S, s_0, T, E_k, \Rightarrow_k)$ is defined below; and (2) $Ch = \{ch_{12}, ch_{21}, ch_{121o}, ch_{212o}\}$ contains four channels: Channels $ch_{12}$ are exactly-once (EO) channels, which can be seen as a multiset (bag) of messages sent from Partner $P_1$ to Partner $P_2$ that have not yet been received at a given time. Any message in these sets can be received. Channels $ch_{121o}$ are exactly-once-in-order (EOIO) channels that can be represented as first-in-first-out message queues. Only the head messages in these channels can be received. A further restriction on channels is that any type of messages can be sent only to one of the channels.

For each $L_k$, the elements $S$, $s_0$, and $T$ are exactly those defined in the global model $G$. $E_k$ is a set of events such that, for each interaction $i \in I$, (1) if $sender(i) = k$ then there is a send event $!i \in E_k$, or else (2) there is a receive event $?i \in E_k$. The transition relation $\Rightarrow_k$ satisfies the following condition: if and only if $(s, i, s') \in \Rightarrow_k$ then $(s, !i, s') \in \Rightarrow_k$ and $(s, ?i, s') \in \Rightarrow_l$ where $l \neq k$.

The semantics of a local model is given as a labeled transition system $LTS(L) = (C, c_0, \Rightarrow_c)$ that captures all possible interleaving behaviors of the model. Every configuration $c \in C$ consists of the local state of each partner and the content in each channel. In particular, the initial configuration $c_0 = (s_0, s_0, \emptyset, \emptyset, \varepsilon, \varepsilon)$ where both partners are in their initial states and all channels are empty. Given two configurations $c_1 = (s_1^1, s_1^2, ms_{12}^{21}, ms_{21}^{21}, mq_{12}^{21})$ and $c_2 = (s_2^1, s_2^2, ms_{21}^{21}, ms_{21}^{21}, mq_{12}^{21})$. The transition $(c_1, e, c_2) \in \Rightarrow_c$ if and only if, for some partner $P_k$, there exists a transition $(s_k^1, e, s_k^2)$ such that (1) the other partner $P_j$ does not move, i.e., $s_j^1 = s_j^2$; (2) if $e = !m$ and $m$ is sent
to $c^e_{k^j}$ then $m s^k_{j} \cup \{m\} = m s^k_{j}$; (3) if $e = \downarrow m$ and $m$ is sent to $c^s_{k^j}$ then $m q^s_{k^j}.m = m q^s_{k^j}$; (4) if $e = \uparrow m$ and $m$ is received from $c^s_{j^k}$ then $m s^k_{j} = m s^k_{j} \cup \{m\}$; (5) if $e = \downarrow m$ and $m$ is received from $c^s_{j^k}$ then $m q^s_{k^j} = m q^s_{k^j}$. Moreover, all unaffected channels remain unchanged. A trace of the local model is a finite or infinite path starting from the initial configuration in the labeled transition system, and can be represented as a sequence of events.

For the time being, we limit the number of communication partners to two. If more partners are allowed, then additional restrictions are needed to ensure that any communication partner may infer its current local state from the exchanges of messages that are visible to this particular partner. Besides, the above definition of choreography models can be extended with extra conditions on message contents or with additional inhibitors that prevent from sending of messages at certain local states. The details of extensions can be found in [3]. This is however out of the scope of this paper.

### III. LOCAL ENFORCABILITY

In the remainder of this paper, we fix a choreography model $M = (G, L)$. Let us consider labelled transition systems $LTS(G)$ and $LTS(L)$ as well as the sets of their traces $Traces(G)$ and $Traces(L)$. Note that while $LTS(G)$ has a finite number of states, $LTS(L)$ may have an infinite number of states if there are no restrictions on the channel size.

Note also that the $G$ and $L$ have different alphabets. In order to connect the two models we map the alphabet of interactions used by $G$ to their corresponding send and receive events in $L$. This can be done by either from the sender or from the receiver point of view. According to it there are two kinds of local enforceability studied in the paper: one from the sender point of view and the other from the receiver point of view.

In [2] we have defined a notion of local enforceability as a trace inclusion $Traces(L) \subseteq Traces(G)$. On the contrary, our definition here is stronger than trace inclusion in general, especially for the case of non-deterministic systems. It is equivalent to trace inclusion only in the case of deterministic systems. Intuitively, apart from trace inclusion, we expect that each sub-trace in $G$ simulating a sub-trace in $L$ can be extended in order to simulate the whole trace. Note that the theoretical results presented in this paper hold for both deterministic and non-deterministic systems. We formally define local enforceability as follows.

Let $r$ be a trace of $L$. We use $r|_s$ (and $r|_r$) to denote the subsequence of $r$ obtained by removing all receive events (and send events resp.). We define that a trace $r_g$ in the global model $G$ send-simulates $r$ if, for any $k$, if the $k$-th event in $r|_s$ is $\downarrow 1$ then the $k$-th interaction in $r_g$ is $i$. Receive-simulation is similarly defined. The model $M$ is said to be send enforceable (local enforceable from the sender point of view) if the following condition is satisfied: For any trace $r$ of $L$, if $r$ leads to a configuration where the send event $\downarrow 1$ is enabled, then for every trace $r_g$ of $G$ that send-simulates $r$, the interaction $i$ must be also enabled at the end of $r_g$. In this case, $L$ is a send-refinement of $G$. Receive-enforceability and receive-refinement are similarly defined.

The previous example shown in Figure 1 is both send and receive enforceable if all messages are sent through EOIO channels.

### IV. INCONSUMABLE MESSAGES

The model $M$ is free of inconsumable messages if, in any reachable configuration of the local model, the event $\downarrow m$ is enabled whenever one of the following conditions is satisfied: (1) a message $m$ is in any of the EO channels, or (2) $m$ is the head message of any of the EOIO channels. Figure 2 shows a model where both $i_1$ and $i_2$ are initiated by a same partner. The execution of the model may incur inconsumable messages if $i_1$ and $i_2$ are both sent to the EO channel: Consider the situation in which a message $i_2$ is sent immediately after a message $i_1$ is sent. The receiver is now at its local state $s_0$, and two messages are available to receive. However, the receiver can only receive the message $i_1$, and the message $i_2$ becomes inconsumable in this moment. On the contrary, if both $i_1$ and $i_2$ are sent to the EOIO channel, the model is free of inconsumable messages.

![Figure 2. A send enforceable model with inconsumable messages.](image-url)

Note that our definition of inconsumable messages does not imply the weaker property that a message can never be received once it was sent. Such definitions were used in different contexts [5]. The motivation for the stronger definition follows directly from architectural decisions which had been taken for the real systems which are the basis of our considerations.

In these systems, as for example the one described in [6], services exchange messages through channels and insert arriving messages into queues. These queues are relatively simply processed by the receiving components, namely by selecting the head of the queue to be processed next. Because there is no inconsumable message (according to our strong definition), the component must be able to process this message. If it cannot do so the protocol or the used channel is faulty.

Consider the case that we had stuck to the weaker definition. Then the receiving component would need to examine the messages in the queue one by one and would need to to pick the first processable one. In theory, no message
could ever be erased from the queue since there will always be states where each message is picked, and so the length of the queue would grow unlimitedly. The higher cost for processing queues would moreover not be compensated by an easier verification process: the temporal weaker property would not be considerably easier to verify than our stronger safety property.

V. CONTAINMENT RELATIONS

Let SE (and RE) denote the set of send (and receive resp.) enforceable models, and AIM denote the set of models free of inconsumable messages. We show the containment relations among these classes in Figure 3. We prove each of the (non-)containment relations in the following.

![Figure 3. Containment relations.](image)

**Theorem 1 (AIM ⊂ SE):** If a model is free of inconsumable messages then it is also send enforceable, but not vice versa.

**Proof:** We first show that $M \in AIM \rightarrow M \in SE$. Using proof by contradiction, assume that $M \in AIM$ and $M \notin SE$. Then there exists a trace $r$ in $L$ such that (1) some event $!m$ is enabled at the end of $r$; (2) there exists a trace $r_g$ in $G$ that send-simulates $r$; and (3) the interaction $m$ is not possible at the end of $r_g$. Without loss of generality, we assume that $sender(m) = 1$, i.e., Partner $P_1$ sends messages $m$. Let $r$ result in a sequence of configurations $c_0, \ldots, c_k$ where, for each $i$, $c_i$ results from the occurrence of the event $e_i$ at $c_{i-1}$. Furthermore, for each $c_i = (s_i^1, s_i^2, ms_i^{21}, ms_i^{21}, mq_i^{21}, mq_i^{21})$, let $s_i$ denote the state in $r_g$ where $c_i$ is reached.

We now construct a trace $r'$ of $L$ such that, during the construction, $r'$ visits a sequence of configurations $c'_0, \ldots, c'_k$ where each configuration $c'_i = (s'_i, s_i, \emptyset, ms_i^{21}, mq_i^{21}, \varepsilon)$, i.e., the channels $ch_{12}^{eo}$ and $ch_{12}^{eo}$ are always empty at $c'_i$, and the contents of both $ch_{21}^{eo}$ and $ch_{21}^{eo}$ in $c'_i$ are the same as in $c_i$. Intuitively, Partner $P_1$ replays its part of the execution in $r$ and Partner $P_2$ replays the trace $r_g$. Note that $r'$ may visit some intermediate configurations between each $c'_i$ and $c'_{i+1}$.

In the beginning, $r'$ is empty and we are at the initial configuration $c'_0 = c_0 = (s_0, s_0, \emptyset, \emptyset, \varepsilon, \varepsilon)$.

Now, suppose that $r'$ has reached $e'_i = (s'_i, s_i, \emptyset, ms_i^{21}, \varepsilon, mq_i^{21})$. We extend $r'$ according to how $r$ is extended from $c_i$ to $c_{i+1}$:

**Case 1:** Assume that the event $e_{i+1}$ between $c_i$ and $c_{i+1}$ in $r$ is $!m'$ where $sender(m') = 1$. Then $s_{i+1}$ must be reached from $s_i$ by the interaction $m'$ in $r_g$. When $m'$ is sent to $ch_{12}^{eo}$, we first extend $r'$ to a configuration $c' = (s'_i, s_i, \{m\}, ms_i^{21}, \varepsilon, mq_i^{21})$ by sending $m'$, then to the configuration $c'' = (s'_i, s_i, \emptyset, ms_i^{21}, \varepsilon, mq_i^{21})$ by receiving $m'$ immediately. Since no channels other than $ch_{12}^{eo}$ are affected by the event $!m'$, we have $ms_i^{21} = ms_{i+1}^{21}$ and $mq_i^{21} = mq_{i+1}^{21}$. This implies $c'_i = c'_i$. The same result holds similarly when $m'$ is sent to $ch_{12}^{eo}$.

**Case 2:** Assume that the event $e_{i+1}$ is $?m'$ where $sender(m') = 2$. Note that in this case $s_i = s_{i+1}$, i.e., Partner $P_1$ does not move. Also, $s_{i+1}$ must be reached from $s_i$ by the interaction $m'$ in $r_g$. When $m'$ is sent to $ch_{21}^{eo}$, we extend $r'$ to a configuration $c' = (s'_i, s_{i+1}, \emptyset, ms_{i+1}^{21} \{m\}, \varepsilon, mq_{i+1}^{21})$, by sending $m'$. Since only $ch_{21}^{eo}$ is affected by the event $!m'$, we have $ms_{i+1}^{21} = ms_{i+1}^{21}$ and all other channels are unchanged. This implies $c'_i = c'_{i+1}$. The same result holds similarly when $m'$ is sent to $ch_{21}^{eo}$.

**Case 3:** Assume that the event $e_{i+1}$ is $?m'$ where $sender(m') = 1$. Note that in this case $s_i = s_{i+1}$ and $s_{i+1} = s_i$. Moreover, since the channels $ch_{12}^{eo}$ and $ch_{21}^{eo}$ are not affected by $?m'$, we have $ms_{i+1}^{21} = ms_{i+1}^{21}$ and $mq_{i+1}^{21} = mq_{i+1}^{21}$. Therefore, $c'_{k+1} = c'_k$ and $r'$ is not extended.

**Case 4:** Assume that the event $e_{i+1}$ is $?m'$ where $sender(m') = 2$. In this case, $s_{i+1} = s_i$. When $m'$ is sent to $ch_{21}^{eo}$, we extend $r'$ to the configuration $c' = (s'_i, s_{i+1}, \emptyset, ms_{i+1}^{21} \{m\}, \varepsilon, mq_{i+1}^{21})$ by receiving $m'$, which is possible due to our induction assumption that the content of $ch_{21}^{eo}$ is same at $c_i$ and $c'_i$. Trivially, the content of $ch_{12}^{eo}$ is still same at $c_{i+1}$ and $c'_{i+1}$ after receiving $m'$, and all other channels are not changed. This implies $c'_i = c'_{i+1}$. The same result similarly holds when $m'$ is sent to $ch_{12}^{eo}$.

After $r'$ reaches $c'_k$, we extend $r'$ to some configuration $c_{k+1}$ by sending $m$. Since Partner $P_2$ does not move in this step, its local state in $c_{k+1}$ is still $s_k$. Moreover, $m$ must be receivable since both $ch_{12}^{eo}$ and $ch_{21}^{eo}$ are empty at $c'_k$. However, Partner $P_2$ cannot receive it because it is at the local state $s_k$. This is an inconsumable message, which contradicts our assumption.

Finally, the model in Figure 2 shows $AIM \neq SE$ since it is send enforceable but has inconsumable messages, if only EO channels are used.

**Theorem 2 (AIM ⊂ RE):** If a model is free of inconsumable messages then it is also receive enforceable, but not vice versa.

**Proof:** The proof is similar to that of Theorem 1. We first show that $M \in AIM \rightarrow M \in RE$. Assume by contradiction that $M \in AIM$ and $M \notin RE$. Then there exists a trace $r$ in $L$ such that (1) some event $?m$ is enabled at the end of $r$; (2) there exists a trace $r_g$ in $G$ that receive-simulates $r$; and (3) the interaction $m$ is not possible at the end of $r_g$. We assume that $sender(m) = 1$. Let $r$ result in a sequence of configurations $c_0, \ldots, c_k$ where, for each $i$, $c_i$ results from the occurrence of the event $e_i$ at $c_{i-1}$. Furthermore, for
each \(c_i = (s_i^1, s_i^2, m_{s_i}^{12}, m_{q_i}^{21}, m_{q_i}^{12})\), let \(s_i\) denote the state in \(r_g\) when \(c_i\) is reached.

We now construct a trace \(r'\) of \(L\) such that, during the construction, \(r'\) visits a sequence of configurations \(c_0', \ldots, c_k'\) (again, there could be some configurations between each \(c_i'\) and \(c_{i+1}'\)) where each configuration \(c_i' = (s_i', s_i, m_{s_i}^{12}, m_{q_i}^{21}, \varepsilon)\), i.e., the channels \(ch_{21}^{eo}\) and \(ch_{21}^{eoi}\) are always empty at \(c_i'\), and the contents of \(ch_{12}^{eo}\) and \(ch_{12}^{eoi}\) in \(c_i'\) are the same as in \(c_i\). Intuitively, Partner \(P_1\) replays its part of the execution in \(r\) and Partner \(P_2\) replays the trace \(r_g\). The strategy here is the same as the one used in the proof of Theorem 1. However, notice that the conditions on channels for the configurations \(c_i'\) are totally different.

In the beginning, \(r'\) is empty and we are at the initial configuration \(c_0' = c_0 = (s_0, s_0, \emptyset, \emptyset, \varepsilon, \varepsilon)\).

Now, suppose that \(r'\) has reached \(c_i' = (s_i', s_i, m_{s_i}^{12}, m_{q_i}^{21}, \varepsilon)\). We extend \(r'\) according to how \(r\) is extended from \(c_i\) to \(c_{i+1}\):

**Case 1:** Assume that the event \(e_{i+1}\) is \(!m'\) where \(\text{sender}(m') = 1\). In this case \(s_{i+1} = s_i\). We can extend \(r'\) to \(c_{i+1}'\) by sending \(m'\). This is guaranteed by the induction assumption that the contents of \(ch_{12}^{eo}\) and \(ch_{12}^{eoi}\) in \(c_i'\) are the same as in \(c_i\). Note that other channels are not affected.

**Case 2:** Assume that the event \(e_{i+1}\) is \(!m'\) where \(\text{sender}(m') = 2\). Note that in this case \(s_{i+1} = s_{i+1}^1\) and \(s_{i+1} = s_i\). Moreover, \(m_{s_i}^{12} = m_{s_{i+1}}^{12}\) and \(m_{q_i}^{21} = m_{q_{i+1}}^{21}\) since \(ch_{21}^{eo}\) and \(ch_{12}^{eoi}\) are not affected by sending \(m'\). This implies \(c_i' = c_{i+1}'\) and \(r'\) is not extended.

**Case 3:** Assume that the event \(e_{i+1}\) is \(?m'\) where \(\text{sender}(m') = 1\). Note that in this case \(s_{i+1} = s_{i+1}^1\) and \(s_{i+1} = s_i\) is reached from \(s_i\) by \(m'\). We extend \(r'\) to \(c_{i+1}'\) by receiving \(m'\), which is possible due to our induction assumption that the contents of \(ch_{12}^{eo}\) and \(ch_{12}^{eoi}\) in \(c_i'\) are the same as in \(c_i\).

**Case 4:** Assume that the event \(e_{i+1}\) is \(?m'\) where \(\text{sender}(m') = 2\). In this case, \(s_{i+1} = s_{i+1}^1\) is reached from \(s_i\) by \(m'\). We extend \(r'\) to \(c_{i+1}'\) by first sending \(m'\) and then receiving \(m'\) immediately. After these two steps, \(ch_{21}^{eo}\) and \(ch_{12}^{eoi}\) are still empty and other channels are not affected.

After \(r'\) reaches \(c_k'\), the message \(m\) must exist in either \(ch_{12}^{eo}\) or \(ch_{12}^{eoi}\) and is ready to be received, this is because the contents of these channels are the same as in \(c_k\). However, since Partner \(P_2\) is in local state \(s_k\), the message \(m\) cannot be received. This is an inconsumable message, which contradicts our assumption.

Finally, the model in Figure 4 shows \(\text{AIM} \not= \text{RE}\). Inconsumable messages occur when both a message \(i1\) and a message \(i2\) are sent. In this case, one partner is at the local state \(s_1\) and unable to receive \(i2\), while the other is at the local state \(s_2\) and unable to receive \(i1\). Both messages are then inconsumable.

**Theorem 3** \((\text{SE} \not\subset \text{RE}\) and \(\text{RE} \not\subset \text{SE})\): A send enforceable model is not necessarily receive enforceable, and vice versa.

**Proof:** The model in Figure 4 is receive enforceable but not send enforceable. The model in Figure 5 is send enforceable but not receive enforceable when all messages are sent to EO channels. This is because the following sequence of events \((?i2, ?i3, ?i1)\) is possible but the trace \((i2, i3, i1)\) is not present in the global model.

**VI. Containment Relations for EOIO Only Models**

Now we consider the class of choreography models in which all messages are sent to EOIO channels. Let \(\text{SE}_{EOIO}\) be the set of send enforceable models in this class, and \(\text{RE}_{EOIO}\) and \(\text{AIM}_{EOIO}\) be similarly defined. The containment relations are shown in Figure 6. Due to space limitation all proofs in this section are included in appendices.

**Theorem 4** \((\text{AIM}_{EOIO} = \text{SE}_{EOIO})\): If an EOIO only model is free of inconsumable message then it is send enforceable, and vice versa.

**Proof:** The direction \(\text{AIM}_{EOIO} \subset \text{SE}_{EOIO}\) has been shown in Theorem 1. It remains to prove \(\text{SE}_{EOIO} \subset \text{AIM}_{EOIO}\). Assume by contradiction that \(M \in \text{SE}_{EOIO}\) and \(M \notin \text{AIM}_{EOIO}\). Then there exists a trace \(r\) of \(L\) that results in a sequence of configurations \(c_0, \ldots, c_k\) such that a message \(m\) is ready to be received but cannot be received at \(c_k\). Suppose \(m\) is sent by Partner \(P_j\) at the configuration \(c_j\). Let \(r_j\) denote the projection of \(r\) that ends at \(c_j\). Let \(c_i = (s_i^1, s_i^2, \emptyset, \emptyset, m_{q_i}^{12}, m_{q_i}^{21})\) for each \(i\).

Let \(r'_j\) denote the projection of \(r\) on the execution of Partner \(P_i\). We construct a trace \(r'_g\) of \(G\) such that it is
equivalent to \( r'_g \) after replacing every event with its corresponding interaction, i.e., \( r'_g \) replays \( P_2 \)'s part of execution in \( r \).

We now construct a trace \( r' \) of \( L \) that can be send-simulated by \( r'_g \). During the construction, \( r' \) visits a sequence of configurations \( c'_0, \ldots, c'_k \) (intermediate configurations possible between each \( c'_i \) and \( c'_{i+1} \)) such that each \( c'_i = (s_{n_i}', s_2^i, 0, 0, \varepsilon, q_i) \) where \( s_{n_i}' \) and \( q_i \) satisfy the following conditions: (*) \( s_{n_i}' \in \{ s_0', \ldots, s_1') \} \) and \( n_i \geq n_{i-1} \); (***) let \( r'_i \) be the prefix of \( r' \) that ends at \( c'_i \), then \( r'_i \) is the longest prefix of \( r_{j_i} \) such that \( q_i \) is the shortest queue of which \( mq_{21}^i \) is a suffix, i.e., Partner \( P_1 \) should receive as many messages as possible before \( c'_i \) is reached. Intuitively, Partner \( P_1 \) replays its part of execution in \( r \) until the local state \( s_{i}' \), i.e., immediately before the sending of \( m_i \), and Partner \( P_2 \) replays its part of execution of \( r \). Furthermore, the channel \( ch_{12}^{zio} \) is always empty at \( c'_i \).

In the beginning, \( r' \) and \( r'_g \) are empty, and we are at the initial configuration \( c'_0 \) and the initial state \( s_0 \). The above properties hold trivially.

Now suppose that \( r' \) has reached \( c'_i = (s_{n_i}', s_2^i, 0, 0, \varepsilon, q_i) \) and \( r'_g \) has reached \( s_2^j \) where the above properties are satisfied. We extend \( r'_g \) according to how \( r'_g \) is extended.

**Case 1:** Assume that \( s_{i+1}' \) is reached from \( s_i' \) by the interaction \( m_i \) where \( sender(m_i) = 2 \). In this case, we first extend \( r' \) to a configuration \( c' = (s_{n_i}', s_{i+1}', 0, 0, \varepsilon, q_i, m'_{i}) \) by sending \( m' \). Then, we extend \( r' \) to the configuration \( c_{i+1}' \) by receiving as many messages as possible from the channel \( ch_{21}^{zio} \) by Partner \( P_1 \). It is obvious that the property (***) is satisfied since the order of sending and receiving messages in \( r \) is preserved in \( r' \). In particular, if \( q_{i+1} \neq mq_{21}^{i+1} \) and \( s_{n_{i+1}}' \neq s_{j}' \) then Partner \( P_1 \) is supposed to send some message at \( s_{n_{i+1}}' \) in \( r \). Otherwise, it should have proceeded to receive more messages.

**Case 2:** Assume that \( s_{i+1}' \) is reached from \( s_i' \) by the interaction \( m_i \) where \( sender(m_i) = 1 \). Then, Partner \( P_1 \) must be able to send the message \( m' \) at \( s_{i+1}' \). Otherwise, \( P_1 \) and \( P_2 \) will wait for messages from each other that can never be available, then the trace \( r \) cannot be possible. We extend \( r' \) by first sending \( m' \) and then receiving it immediately. This results in a configuration \( c' = (s_{n_i+1}', s_{i+1}', 0, 0, \varepsilon, q_i, q_k) \). Then, we further extend \( r' \) to receive as many messages as possible from \( ch_{21}^{eoio} \) to reach the configuration \( c_{i+1}' \).

By the end of the construction, \( r' \) reaches the configuration \( c_k' = (s_{j_k}', s_2^k, 0, 0, \varepsilon, q_k) \) where all messages sent by \( P_1 \) before \( m \) have been received by \( P_2 \). Note that \( P_2 \)'s execution in \( r' \) does not rely on any messages sent by \( P_1 \) after \( m \). Now we extend \( r' \) by sending \( m \) to the configuration \( c_{k+1}' = (s_{j_k+1}', s_2^k, 0, 0, (m), q_k) \). However, the send-simulating trace \( r'_g \) cannot be extended by the interaction \( m \). Otherwise, the message must be receivable by \( P_2 \) at \( s_2^k \). This contradicts our assumption.

The proof of Theorem 2 shows that \( AIM_{EOIO} \subseteq RE_{EOIO} \). In particular, the example in Figure 4, which we used to show \( AIM \neq RE \), makes no assumption on the usages of EO or EOIO channels.

Theorem 5 (SE_{EOIO} \subseteq RE_{EOIO}): If an EOIO only model is send enforceable then it is also receive enforceable, but not vice versa.

Proof: Due to limited space, we give only the intuition of the proof here. Note that, for any trace \( r \) of \( L \), the subsequences \( r|_s \) and \( r|_r \) preserve the same order of messages sent by one same partner. So, when the order of two messages \( a \) and \( b \) are different in \( r|_s \) and \( r|_r \), they must be sent by different partners. In this case, we can switch the order of the sending of these two messages. We repeat this procedure until we obtain a new trace \( r' \) where \( r'|_r \) is a prefix of \( r'|_s \). This implies that if \( r' \) can be send-simulated then it can also be receive-simulated. Note that new order mismatches may be introduced by switching two sending events. However, the newly introduced mismatches always occur closer and closer to the initial configurations. This guarantees the termination of the event switching procedure.

Finally, the example in Figure 4 shows \( SE_{EOIO} \neq RE_{EOIO} \).

VII. Related Work

In this paper we use the choreography modeling language MCM [4, 2]. There also exist other choreography languages like WSCDL [7], BPMN [8], or Let’s Dance [9], where the interaction protocols between a set of loosely coupled components communicating over message channels is described from the perspective of a global observer. The study of relations between local enforceability and inconsumable messages for these languages remains a matter of future work.

The semantics of the MCM language is given on the basis of labelled transition systems. The formal analysis of LTS-based systems has been profoundly studied in the areas of both model checking [10] and theorem proving [11].

The notion of local enforceability that we use in this paper corresponds in some sense to the one proposed in [12], [13]: “The global model can be mapped into local ones in such a way that the resulting local models satisfy the following two conditions: (i) they contain only interactions described in the global model; and (ii) they are able to collectively enforce all the constraints expressed in the global model.” In our case local partners use exactly the interactions and constraints from the global model. Checking local enforceability means to check if the obtained local model does not show additional (undesired) behaviour compared to the global model. Our notion of local enforceability builds upon the concepts of trace inclusion and simulation [14]. Note that trace simulation is different than the traditional concept of state-based simulation as introduced in process algebra [15].
VIII. CONCLUSION

This paper addresses the formal verification of message choreography models, as an important type of model for modern enterprise applications, at the example of the MCM approach developed at SAP Research. The relation among two significant properties of choreography models, namely local enforceability and absence of inconsumable messages are studied. We arrive at, among others, an important conclusion that, whenever a choreography model has no inconsumable messages, it is also local enforceable no matter how we connect the behavior of the global model to the behavior of the local partner model. This is very useful because the mechanical checking of inconsumable messages is much less costly than that of local enforceability: For a model that possesses a finite state space, the checking of inconsumable messages can be accomplished by an exhaustive exploration of the state space of only the local partner model whose runtime complexity is linear in the size of the state space. On the contrary, the checking of local enforceability needs essentially to construct a simulation relation between the traces of the global model and the traces of the local partner model, which requires in general exponential time.

In future work we will consider various extensions of the choreography language, including inhibitors and constraints on message contents, and study their impacts on the containment relations between the two above mentioned properties. Moreover, we mentioned that our definition of local enforceability is stronger than the usual interpretation of simulations as trace inclusion. Therefore, it remains to be an interesting subject how our definition relates to other known classes of simulations. Finally, we will also consider other important properties for MCM such as deadlock freedom.

Acknowledgment. This work was partially supported by the EC-funded project Deploy\textsuperscript{2} (grant no. 214158). We also thank Sebastian Wieczorek for insightful discussions.

REFERENCES


http://www.deploy-project.eu