A Logic for Choreographies*

Marco Carbone Davide Grohmann Thomas T. Hildebrandt Hugo A. López
IT University of Copenhagen, Rued Langgaards Vej 7, 2300 København S, Denmark
{carbonem,davg,hilde,lopez}@itu.dk

We explore logical reasoning for the global calculus, a coordination model based on the notion of choreography, with the aim to provide a methodology for specification and verification of structured communications. Starting with an extension of Hennessy-Milner logic, we present the global logic ($\mathscr{GL}$), a modal logic describing possible interactions among participants in a choreography. We illustrate its use by giving examples of properties on service specifications. Finally, we show that, despite $\mathscr{GL}$ is undecidable, there is a significant decidable fragment which we provide with a sound and complete proof system for checking validity of formulae.

1 Introduction

Due to the continuous growth of technologies, software development is recently shifting its focus on communication, giving rise to various research efforts for proposing new methodologies dealing with higher levels of complexity. A new software paradigm, known as *choreography*, has emerged with the intent to ease programming of communication-based protocols. Intuitively, a choreography is a description of the global flow of execution of a system where the software architect just describes which and in what order interactions *can* take place. This idea differs from the standard approach where the communication primitives are given for each single entity separately. A good illustration can be seen in the way a soccer match is planned: the coach has an overall view of the team, and organises (a priori) how players will interact in each play (the rôle of a choreography); once in the field, each player performs his role by interacting with each of the members of his team by throwing/receiving passes. The way each player synchronise with other members of the team represents the rôle of an orchestration.

The work in [4] formalises the notion of choreography in terms of a calculus, dubbed the *global calculus*, which pinpoints the basic features of the choreography paradigm. Although choreography provides a good abstraction of the system being designed allowing to *forget* about common problems that can arise when programming communication (e.g. races over a channel), it can still have complex structures hence being often error prone. Additionally, choreography can be non-flexible in early design stages where the architect might be interested in designing only parts of a system as well as specifying only parts of a protocol (e.g. initial and final interactions). In this view, we believe that a logical approach can allow for more modularity in designing systems e.g. providing partial specification of a system using the choreography paradigm.

In order to illustrate the approach proposed in this work, let us consider an online booking scenario. On one side, consider an airline company AC which offers flights directly from its website. On the other side, there is a customer looking for the best offers. We can informally describe the interaction protocol in terms of a sequence of allowed interactions (as in a choreography) as follows:

*The authors are listed in alphabetical order.

K. Honda and A. Mycroft (Eds.): Programming Language Approaches to Concurrency and communication-cEntric Software 2010 (PLACES'10)
EPTCS 69, 2011, pp. 29–43, doi:10.4204/EPTCS.69.3

1. Customer establishes a communication with AC;
2. Customer asks AC for a flight proposal given a set of constraints;
3. AC establishes a communication with partner AC' serving the destination asked by the costumer;
4. AC forwards the request made by the customer;
5. AC' sends an offer to AC;
6. AC forwards the offer to the customer

Note that each step above represents a communication. In the same way that a choreographical specification describes each of the interactions between participants, a logical characterisation of choreographies denotes formulae describing the evolution of such interactions. However, a logical characterisation gives extra flexibility to the specification of interactions: When writing a logical property describing specific communication patterns we focus on describing **only** the sequence of *key interactions*, leaving room for implementations that include extra behaviour that does not compromise the fulfilment of the property. For instance, in the above example, one can describe a property leaving out the details on the forward of the request to the airline partner, in a statement like *"given an interaction between the customer and AC featuring a booking request, then there is an eventual response directed to the customer with an offer matching the original session"* (in this case, the offer is not necessarily from the airline originally contacted but from one of its partners).

In this document, we provide a link between choreographies and logics. Starting with an extension of Hennessy-Milner logic [10], we provide the syntax and the semantics of a logic for the global calculus as well as several examples of choreographical properties. On decidability issues, we found out that the whole set of the logic is undecidable on the global calculus with recursion. As a result, we focus our studies in a decidable fragment, providing a proof system that allows for property verification of choreographies and show that it is sound and complete, in the sense that all and only valid formulae specified in the global logic can be provable in the proof system. Moreover, we can conclude that the proof checking algorithm using this proof system is terminating.

Overview of the document First, in Section 2 we recall the formal foundations of the global calculus, and equip it with a labelled transition semantics. A logic characterisation of the calculus and several examples of the use of the logic are presented in Section 3. We proceed with the study of undecidability for the logic in Section 4, and a proof system relating the logical characterisation and the global calculus for a decidable fragment of the language is presented in Section 5. Finally, concluding remarks are presented in Section 6.

2 The Global Calculus

The Global Calculus (GC) [4, 5] originates from the Web Service Choreography Description Language (WS-CDL) [12], a description language for web services developed by W3C. Terms in GC describe choreographies as interactions between participants by means of message exchanges. The description of such interactions is centred on the notion of a *session*, in which two interacting parties first establish a private connection via some public channel and then interact through it, possibly interleaved with other sessions. More concretely, an interaction between two parties starts by the creation of a fresh session identifier, that later will be used as a private channel where meaningful interactions take place. Each session is fresh and unique, so each communication activity will be clearly separated from other interactions. In this section, we provide an operational semantics for GC in terms of a label transition

systems (LTS) [16] describing how global descriptions evolve, and relate to the type discipline that describes the structured sequence of message exchanges between participants from [5].

2.1 Syntax

Let $\mathscr{C}, \mathscr{C}', \ldots$ denote *terms* of the calculus, often called *interactions* or *choreographies*; $A, B, C, \ldots$ range over *participants*; $k, k', \ldots$ are *linear channels*; $a, b, c, \ldots$ *shared channels*; $v, w, \ldots$ variables; $X, Y, \ldots$ process variables; $l, l_i, \ldots$ labels for branching; and finally $e, e', \ldots$ over unspecified arithmetic and other first-order expressions. We write $e@A$ to mean that the expression e is evaluated using the variable related to participant A in the store.

Definition 2.1. *The syntax of the global calculus [4] is given by the following grammar:*

$$
\begin{array}{rll}
\mathscr{C} ::= & \mathbf{0} & \text{(inaction)} \\
\mid & A{\to}B{:}a(k).\,\mathscr{C} & \text{(init)} \\
\mid & A{\to}B : k\langle e, y\rangle.\,\mathscr{C} & \text{(com)} \\
\mid & A{\to}B{:}k[l_i : \mathscr{C}_i]_{i\in I} & \text{(choice)} \\
\mid & \mathscr{C}_1 \mid \mathscr{C}_2 & \text{(par)} \\
\mid & \mathbf{if}\, e@A\, \mathbf{then}\, \mathscr{C}_1\, \mathbf{else}\, \mathscr{C}_2 & \text{(cond)} \\
\mid & X & \text{(recvar)} \\
\mid & \mu X.\, C & \text{(recursion)}
\end{array}
$$

Intuitively, the term (inaction) denotes a system where no interactions take place. (init) denotes a session initiation by A via B's service channel a, with a fresh session channel k and continuation $\mathscr{C}$. Note that k is bound in $\mathscr{C}$. (com) denotes an in-session communication of the evaluation (at A's) of the expression e over a session channel k. In this case, y does not bind in $\mathscr{C}$ (our semantics will treat y as a variable in the store of B). (choice) denotes a labelled choice over session channel k and set of labels I. In (par), $\mathscr{C}_1 \mid \mathscr{C}_2$ denotes the parallel product between $\mathscr{C}_1$ and $\mathscr{C}_2$. (cond) denotes the standard conditional operator where $e@A$ indicates that the expression e has to be evaluated in the store of participant A. In (recursion), $\mu X.\,\mathscr{C}$ is the minimal fix point operation for recursion, where the variable X of (recvar) is bound in $\mathscr{C}$. The free and bound session channels and term variables are defined in the usual way. The calculus is equipped with a standard structural congruence $\equiv$, defined as the minimal congruence relation on interactions $\mathscr{C}$, such that $\equiv$ is a commutative monoid with respect to $\mid$ and $\mathbf{0}$, it is closed under alpha equivalence $\equiv_\alpha$ of terms, and it is closed under the recursion unfolding, i.e., $\mu X.\mathscr{C} \equiv \mathscr{C}[\mu X.\mathscr{C}/X]$.

Remark 2.2 (Differences with the approach in [5]). Excluding the lack of local assignment, we argue that this monadic version of GC is, to some extent, as expressive as the one Global Calculus originally reported in [5]. In particular, note that $A{\to}B : k\langle \mathsf{op}, e, y\rangle$ in [5] captures both selection and message passing which are instead disentangled in our case (mainly for clarity reasons). The absence of op in the interaction process $A{\to}B : k\langle e, y\rangle$ can be easily encoded with the existing operators. In fact, $\Sigma_{i\in I} A{\to}B : k\langle op_i, e, y\rangle.\,\mathscr{C}'_i$ can be decomposed into $A{\to}B{:}k[op_i : \mathscr{C}''_i]_{i\in I}$ where $\mathscr{C}''_i = A{\to}B : k\langle e, y\rangle.\,\mathscr{C}'_i$ (although we lose atomicity).

2.2 Semantics

We give the operational semantics in terms of configurations $(\sigma, \mathscr{C})$, where σ represents the state of the system and $\mathscr{C}$ the choreography actually being executed. The state σ contains a set of variables

$$(\text{G-INIT})\ \frac{h \text{ fresh}}{(\sigma, A{\to}B{:}a(k).\ \mathscr{C}) \xrightarrow{\text{init } A\to B \text{ on } a(h)} (\sigma, \mathscr{C}[h/k])}$$

$$(\text{G-COM})\ \frac{\sigma(e@A) \Downarrow v}{(\sigma, A{\to}B : k\langle e,x\rangle.\ \mathscr{C}) \xrightarrow{\text{com } A\to B \text{ over } k} (\sigma[x@B \mapsto v], \mathscr{C})}$$

$$(\text{G-CHOICE})\ \frac{}{(\sigma, A{\to}B{:}k[l_i : \mathscr{C}_i]_{i\in I}) \xrightarrow{\text{sel } A\to B \text{ over } k:l_i} (\sigma, \mathscr{C}_i)}$$

$$(\text{G-PAR})\ \frac{(\sigma, \mathscr{C}_1) \xrightarrow{\ell} (\sigma', \mathscr{C}_1')}{(\sigma, \mathscr{C}_1 \mid \mathscr{C}_2) \xrightarrow{\ell} (\sigma', \mathscr{C}_1' \mid \mathscr{C}_2)}$$

$$(\text{G-STRUCT})\ \frac{\mathscr{C} \equiv \mathscr{C}' \quad (\sigma, \mathscr{C}') \xrightarrow{\ell} (\sigma', \mathscr{C}'') \quad \mathscr{C}'' \equiv \mathscr{C}'''}{(\sigma, \mathscr{C}) \xrightarrow{\ell} (\sigma', \mathscr{C}''')}$$

$$(\text{G-IFT})\ \frac{\sigma(e@A) \Downarrow \texttt{tt} \quad (\sigma, \mathscr{C}_1) \xrightarrow{\ell} (\sigma', \mathscr{C}_1')}{(\sigma, \textbf{if}\, e@A\, \textbf{then}\, \mathscr{C}_1\, \textbf{else}\, \mathscr{C}_2) \xrightarrow{\ell} (\sigma', \mathscr{C}_1')}$$

$$(\text{G-IFF})\ \frac{\sigma(e@A) \Downarrow \texttt{ff} \quad (\sigma, \mathscr{C}_2) \xrightarrow{\ell} (\sigma', \mathscr{C}_2')}{(\sigma, \textbf{if}\, e@A\, \textbf{then}\, \mathscr{C}_1\, \textbf{else}\, \mathscr{C}_2) \xrightarrow{\ell} (\sigma', \mathscr{C}_2')}$$

Table 1: Operational Semantics for the Global Calculus

labelled by participants. As described in the previous subsection, a variable x located at participant A is written as $x@A$. The same variable name labelled with different participant names denotes different variables (hence $\sigma(x@A)$ and $\sigma(x@B)$ may differ). Formally, the operational semantics is defined as a labelled transition system (LTS). A transition $(\sigma, \mathscr{C}) \xrightarrow{\ell} (\sigma', \mathscr{C}')$ says that a choreography $\mathscr{C}$ in a state σ executes an action (or label) ℓ and evolves into $\mathscr{C}'$ with a new state σ'. Actions are defined as $\ell = \{\mathsf{init}\ A \to B\ \mathsf{on}\ a(k), \mathsf{com}\ A \to B\ \mathsf{over}\ k, \mathsf{sel}\ A \to B\ \mathsf{over}\ k : l_i\}$, denoting initiation, in-session communication and branch selection, respectively. We write $(\sigma, \mathscr{C}) \longrightarrow (\sigma', \mathscr{C}')$ when ℓ irrelevant, and $\longrightarrow^*$ denotes the transitive closure of $\longrightarrow$. The transition relation $\longrightarrow$ is defined as the minimum relation on pairs state/interaction satisfying the rules in Table 1.

Intuitively, transition (G-INIT) describes the evolution of a session initiation: after A initiates a session with B on service channel a, A and B share the fresh channel h locally. (G-COM) describes the main interaction rule of the calculus: the expression e is evaluated into v in the A-portion of the state σ and then assigned to the variable x located at B resulting in the new state $\sigma[x@B \mapsto v]$. (G-CHOICE) chooses the evolution of a choreography resulting from a labelled choice over a session key k. (G-IFT) and (G-IFF) show the possible paths that a deterministic evolution of a choreography can produce. (G-PAR) and (G-STRUCT) behave as the standard rules for parallel product and structural congruence, respectively.

Remark 2.3 (Global Parallel). Parallel composition in the global calculus differs from the notion of parallel found in standard concurrency models based on input/output primitives [14]. In the latter, a term $P_1 \mid P_2$ may allow *interactions* between P_1 and P_2. However, in the global calculus, the parallel composition of two choreographies $\mathscr{C}_1 \mid \mathscr{C}_2$ concerns two parts of the described system where *interactions* may occur in $\mathscr{C}_1$ and $\mathscr{C}_2$ but never across the parallel operator $\mid$. This is because an interaction $A \to B \ldots$ abstracts from the actual end-point behaviour, i.e., how A sends and B receives. In this model, dependencies between two choreographies can be expressed by using variables in the state σ.

In its original presentation [5], GC comes equipped with a reduction semantics unlike the one presented in Table 1. Our LTS semantics has the advantage of allowing to observe changes on the behaviour of the system, which will prove useful when relating to the logical characterisation in Section 3. We conjecture that our proposed LTS semantics and the reduction semantics of the global calculus originally presented in [5] coincide (taking into account the considerations in Remark 2.2).

Example 2.4 (Online Booking). We consider the example presented in the introduction, i.e., a simplified version of the on-line booking scenario presented in [13]. Here, the customer (Cust) establishes a session with the airline company (AC) using service (on-line booking, shorted as ob) and creating the session key k_1. Once the session is established, the customer will request the company about a flight offer with his booking data, along the session key k_1. The airline company will process the customer request and, after requesting another airline company (AC') for the service, will send a reply back with an offer. The customer will eventually accept the offer, sending back an acknowledgment to the airline company using k_1. The following specification in the GC represents the protocol:

$$\begin{aligned}\mathscr{C}_{\mathsf{OB}} = {} & \text{Cust}\to\text{AC:ob}(k_1).\ \text{Cust}\to\text{AC} : k_1\langle \text{booking}, x\rangle.\ \text{AC}\to\text{AC':ob}(k_2). \\ & \text{AC}\to\text{AC'} : k_2\langle \text{x}, x'\rangle.\ \text{AC'}\to\text{AC} : k_2\langle \text{offer}, y\rangle.\ \text{AC}\to\text{Cust} : k_1\langle \text{y}, y''\rangle.\ \text{Cust}\to\text{AC} : k_1\langle \text{accept}, z\rangle.\ \mathbf{0}\end{aligned} \tag{OB}$$

2.3 Session Types for the Global Calculus

We use a generalisation of session types [11] for global interactions, first presented in [5]. Session types in GC are used to structure sequence of message exchanges in a session. Their syntax is as follows:

$$\alpha = \uparrow(\theta).\alpha \mid \downarrow(\theta).\alpha \mid \&\{l_i : \alpha_i\}_{i\in I} \mid \oplus\{l_i : \alpha_i\}_{i\in I} \mid \texttt{end} \mid \mu\mathbf{t}.\ \alpha \mid \mathbf{t} \tag{1}$$

where $\theta, \theta', \ldots$ range over value types $\texttt{bool}, \texttt{string}, \texttt{int}, \ldots$. $\alpha, \alpha', \ldots$ are session types. The first four types are associated with the various communication operations. $\downarrow(\theta).\alpha$ and $\uparrow(\theta).\alpha$ are the input and output types respectively. Similarly, $\&\{l_i : \alpha_i\}_{i\in I}$ is the branching type while $\oplus\{l_i : \alpha_i\}_{i\in I}$ is the selection type. The type `end` indicates session termination and is often omitted. $\mu\mathbf{t}.\ \alpha$ indicates a recursive type with $\mathbf{t}$ as a type variable. $\mu\mathbf{t}.\ \alpha$ binds the free occurrences of $\mathbf{t}$ in α. We take an *equi-recursive* view on types, not distinguishing between $\mu\mathbf{t}.\ \alpha$ and its unfolding $\alpha[\mu\mathbf{t}.\ \alpha/\mathbf{t}]$.

A typing judgment has the form $\Gamma \vdash \mathscr{C} : \Delta$, where Γ, Δ are *service type* and *session type* environments, respectively. Typically, Γ contains a set of type assignments of the form $a@A : \alpha$, which says that a service a located at participant A may be invoked and run a session according to type α. Δ contains type assignments of the form $k[A,B] : \alpha$ which says that a session channel k identifies a session between participants A and B and has session type α when seen from the viewpoint of A. The typing rules are omitted, and we refer to [6] for the full account of the type discipline noting that the observations made in Remark 2.2 will require extra typing rules.

Returning to the specification (OB) in Example 2.4, the service type of the airline company AC at channel *ob* can be described as:

$$\text{ob@AC} : (k_1, k_2)\ k_1 \downarrow \text{booking}(\texttt{string}).\ k_2 \uparrow \text{x}(\texttt{string}).\ k_2 \downarrow \text{offer}(\texttt{int}).\ k_1 \uparrow \text{y}(\texttt{int}).\ k_1 \downarrow \text{accept}(\texttt{int}).\ \texttt{end}.$$

$$
\begin{array}{rlr}
\phi,\chi ::= & \exists t.\ \phi & \text{(f-exists)} \\
| & \phi \wedge \chi & \text{(f-and)} \\
| & \neg\phi & \text{(f-neg)} \\
| & \langle \ell \rangle \phi & \text{(f-action)} \\
| & \mathsf{end} & \text{(f-termination)} \\
| & e_1@A = e_2@B & \text{(f-equality)} \\
| & \phi \mid \chi & \text{(f-parallel)} \\
| & \Diamond\phi & \text{(f-may)}
\end{array}
\qquad
\begin{array}{rlr}
\ell ::= & \mathsf{init}\ A \to B\ \mathsf{on}\ a(k) & \text{(l-init)} \\
| & \mathsf{com}\ A \to B\ \mathsf{over}\ k & \text{(l-com)} \\
| & \mathsf{sel}\ A \to B\ \mathsf{over}\ k : l & \text{(l-branch)}
\end{array}
$$

Table 2: $\mathscr{GL}$: Syntax of formulae

Assumption 2.5. In the sequel, we only consider choreographies that satisfy the typing discipline.

3 $\mathscr{GL}$: A Logic for the Global Calculus

In this section, we introduce a logic for choreographies, inspired by the modal logic for session types presented in [1]. The logical language comprises assertions for equality and value/name passing.

3.1 Syntax

The grammar of assertions is given in Table 2. Choreography assertions (ranged over by $\phi, \phi', \chi, \ldots$) give a logical interpretation of the global calculus introduced in the previous section. The logic includes the standard First Order Logic (FOL) operators $\wedge$, $\neg$, and $\exists$. In $\exists t.\ \phi$, the variable t is meant to range over service and session channels, participants, labels for branching and basic placeholders for expressions. Accordingly, it works as a binder in ϕ. In addition to the standard operators, the operator (f-action) represents the execution of a labelled action ℓ followed by the assertion ϕ. Those labels in ℓ match the ones in the LTS of GC, i.e., they are (l-init), (l-com), and (l-branch). The formula (f-termination) represents the process termination. We also include an unspecified, but decidable, (f-equality) operator on expressions as in [1]. (f-may) denotes the standard eventually operators from Linear Temporal Logic (LTL) [9]. The spatial operator (f-parallel) denotes composition of formulae: because of the unique nature of parallel composition in choreographies, we have used the symbol $\mid$ (as in separation logic [18] and spatial logic [3]) in order to stress the fact that there is no interference between two choreographies running in parallel.

Notation 3.1 (Existential quantification over action labels). In order to simplify the readability, we introduce the concept of existential quantification over action labels as a short-cut to mean the following:

$$
\begin{array}{rcl}
\exists \ell.\ \langle \ell \rangle \phi & \stackrel{\text{def}}{=} & \exists A,B,a,k.\ \langle \mathsf{init}\ A \to B\ \mathsf{on}\ a(k) \rangle.\ \phi \vee \\
& & \exists A,B,k.\ \langle \mathsf{com}\ A \to B\ \mathsf{over}\ k \rangle.\ \phi \vee \\
& & \exists A,B,k,l.\ \langle \mathsf{sel}\ A \to B\ \mathsf{over}\ k : l \rangle.\ \phi\,.
\end{array}
$$

Remark 3.2 (Derived Operators). We can get the full account of the logic by deriving the standard set of strong modalities from the above presented operators. In particular, we can encode the constant true

(tt) and false (ff), the next ($\circ\phi$) and the always operators ($\Box\phi$) from LTL.

$$\texttt{tt} \stackrel{\text{def}}{=} (0@A = 0@A) \qquad \texttt{ff} \stackrel{\text{def}}{=} (0@A = 1@A) \qquad (e_1 \neq e_2) \stackrel{\text{def}}{=} \neg(e_1 = e_2)$$
$$\forall x.\, \phi \stackrel{\text{def}}{=} \neg\exists x.\, \neg\phi \qquad \phi \vee \chi \stackrel{\text{def}}{=} \neg(\neg\phi \wedge \neg\chi) \qquad \phi \Rightarrow \chi \stackrel{\text{def}}{=} \neg\phi \vee \chi$$
$$\Box\phi \stackrel{\text{def}}{=} \neg\Diamond\neg\phi \qquad [\ell]\phi \stackrel{\text{def}}{=} \neg\langle\ell\rangle\neg\phi \qquad \circ\phi \stackrel{\text{def}}{=} \exists\ell.\, \langle\ell\rangle\phi\,.$$

In the rest of this section, we illustrate the expressiveness of our logic through a sequence of simple, yet illuminating examples, giving an intuition of how the modalities introduced plus the existential operator $\exists$ allow to express properties of choreographies.

Example 3.3 (Availability, Service Usage and Coupling). The logic above allows to express that, given a service invoker (known as A in this setting) requesting the service a, there exists another participant (called B in the example) providing a with A invoking it. This can be formulated in $\mathscr{GL}$ as follows:

$$\exists B.\, \langle \mathsf{init}\ A \to B\ \mathsf{on}\ a(k)\rangle\texttt{tt}\,.$$

Assume now, that we want to ensure that services available are actually used. We can use the dual property for availability, i.e., for a service provider B offering a, there exists someone invoking a:

$$\exists A.\, \langle \mathsf{init}\ A \to B\ \mathsf{on}\ a(k)\rangle\texttt{tt}\,.$$

Verifying that there is a service pairing two different participants in a choreography can be done by existentially quantifying over the shared channels used in an initiation action. A formula in $\mathscr{GL}$ representing this can be the following one:

$$\exists a.\, \langle \mathsf{init}\ A \to B\ \mathsf{on}\ a(k)\rangle\texttt{tt}\,.$$

Example 3.4 (Causality Analysis). The modal operators of the logic can be used to perform studies of the causal properties that our specified choreography can fulfil. For instance, we can specify that given an expression e evaluated to true at participant A, there is an eventual firing of a choreography that satisfies property ϕ_1, whilst ϕ_2 will never be satisfied. Such a property can be specified as follows:

$$(e@A = \texttt{tt}) \wedge \Diamond(\phi_1) \wedge \Box\neg\phi_2\,.$$

An interesting aspect of our logic is that it allows for the declaration of partial specification properties regarding the interaction of the participants involved in a choreography. Take for instance the interaction diagram in Figure 1. The participant A invokes service b at B's and then B invokes D's service d. At this point, D can send the content of variable x to A in two different ways: either by using those originally established sessions or by invoking a new service at A's. However, at the end of both computation paths, variable z (located at A's) will contain the value of x. In the global calculus, this two optional behaviour can be modelled as follows:

$$C_1 = A{\to}B{:}b(k).\, B{\to}D{:}d(k').\, D{\to}B : k'\langle x, y_B\rangle.\, B{\to}A : k\langle y_B, z\rangle.\, \mathbf{0} \qquad \text{(Option 1)}$$
$$C_2 = A{\to}B{:}b(k).\, B{\to}D{:}d(k').\, D{\to}A{:}a(k'').\, D{\to}A : k''\langle x, z\rangle\,.\, \mathbf{0}\,. \qquad \text{(Option 2)}$$

We argue that, under the point of view of A, both options are sufficiently good if, after an initial interaction with B is established, there is an eventual response that binds variable z. Such a property can be expressed by the $\mathscr{GL}$ formula:

$$\exists X, k''.\, \langle \mathsf{init}\ A \to B\ \mathsf{on}\ a(k)\rangle\Diamond\Big(\langle \mathsf{com}\ X \to A\ \mathsf{over}\ k''\rangle(z@A = x@D)\Big).\, \texttt{end}\,.$$

Example 3.5 (Response Abstraction).

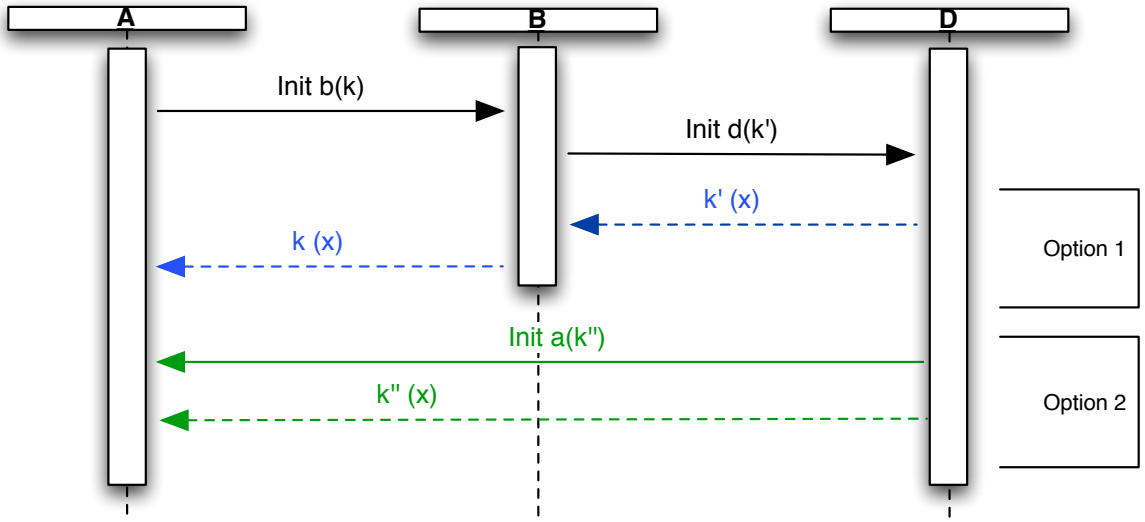

Figure 1: Diagram of a partial specification.

Notice that both the choreographies (Option 1) and (Option 2) *satisfy* the partial specification above. This will be clear in Section 3.2 where we introduce the semantics of logic.

Also note that a third option for the protocol at hand is to use *delegation* (the ability of communicating session keys to third participants not involved during session initiation). However, the current version of the global calculus does not feature such an operation and we leave it as future work.

Example 3.6 (Connectedness). The work in [5] proposes a set of criteria for guaranteeing a safe end-point projection between global and local specifications (note that the choreography in the previous example does not respect such properties). Essentially, a valid global specification has to fulfil three different criteria, namely Connectedness, Well-threadedness and Coherence. It is interesting to see that some of these criteria relate to global and local causality relations between the interactions in a choreography, and can be easily formalised as properties in the choreography logic presented here. Below, we consider the notion of connectedness and leave the other cases as future work. Connectedness dictates a global causality principle among interactions: any two consecutive interactions $\ldots A \to B.\ C \to D \ldots$ in a choreography are such that $B = C$. In the following, let $\mathsf{Interact}(A,B)\phi$ be true whenever $\langle \ell \rangle \phi$ holds for some ℓ with an interaction from A to B. Connectedness can be specified as:

$$\forall A,B.\ \Box\Big(\mathsf{Interact}(A,B)\mathtt{tt} \Rightarrow \exists C.\ \big(\mathsf{Interact}(A,B)\mathsf{Interact}(B,C)\mathtt{tt} \vee \mathsf{Interact}(A,B)\neg\exists \ell \langle \ell \rangle \mathtt{tt}\big)\Big).$$

3.2 Semantics

We now give a formal meaning to the assertions introduced above with respect to the semantics of the global calculus introduced in the previous section. In particular, we introduce the notion of satisfaction. We write $\mathscr{C} \models_\sigma \phi$ whenever a state σ and a choreography $\mathscr{C}$ satisfy a $\mathscr{GL}$ formula ϕ. The relation $\models_\sigma$ is defined by the rules given in Table 3. In the $\exists t.\ \phi$ case, w should be an appropriate value according to the type of t, e.g., a participant if t is a participant placeholder.

Definition 3.7 (Satisfiability, Validity and Logical Equivalence).

$$
\begin{array}{lcl}
\mathscr{C} \models_\sigma \mathsf{end} & \stackrel{\text{def}}{\iff} & \mathscr{C} \equiv \mathbf{0} \\
\mathscr{C} \models_\sigma (e_1@A = e_2@B) & \stackrel{\text{def}}{\iff} & \sigma(e_1@A) \Downarrow v \text{ and } \sigma(e_2@B) \Downarrow v \\
\mathscr{C} \models_\sigma \langle \ell \rangle \phi & \stackrel{\text{def}}{\iff} & (\sigma, \mathscr{C}) \xrightarrow{\ell} (\sigma', \mathscr{C}') \text{ and } \mathscr{C}' \models_{\sigma'} \phi \\
\mathscr{C} \models_\sigma \phi \wedge \chi & \stackrel{\text{def}}{\iff} & \mathscr{C} \models_\sigma \phi \text{ and } \mathscr{C} \models_\sigma \chi \\
\mathscr{C} \models_\sigma \neg\phi & \stackrel{\text{def}}{\iff} & \mathscr{C} \not\models_\sigma \phi \\
\mathscr{C} \models_\sigma \exists t.\, \phi & \stackrel{\text{def}}{\iff} & \mathscr{C} \models_\sigma \phi[w/t] \text{ (for some appropriate } w) \\
\mathscr{C} \models_\sigma \Diamond\phi & \stackrel{\text{def}}{\iff} & (\sigma, \mathscr{C}) \longrightarrow^* (\sigma', \mathscr{C}') \text{ and } \mathscr{C}' \models_{\sigma'} \phi \\
\mathscr{C} \models_\sigma \phi \mid \chi & \stackrel{\text{def}}{\iff} & \mathscr{C} \equiv \mathscr{C}_1 \mid \mathscr{C}_2 \text{ such that } \mathscr{C}_1 \models_\sigma \phi \text{ and } \mathscr{C}_2 \models_\sigma \chi
\end{array}
$$

Table 3: Assertions of the Choreography Logic

- *A formula ϕ is* satisfiable *if there exists some configuration under which it is true, that is, $\mathscr{C} \models_\sigma \phi$ for some $(\mathscr{C}, \sigma)$.*
- *A formula ϕ is* valid *if it is true in every configuration, that is, $\mathscr{C} \models_\sigma \phi$ for every $(\sigma, \mathscr{C})$.*
- *A formula χ is a* logical consequence *of a formula ϕ (or ϕ* logically implies *χ), denote with an abuse of notation as $\phi \models \chi$, if every configuration $(\sigma, \mathscr{C})$ that makes ϕ true also makes χ true.*
- *We say that a formula ϕ is* logical equivalent *to a formula χ, written $\phi \equiv_\models \chi$, if $\phi \models \chi$ iff $\chi \models \phi$.*

4 Undecidability of Global Logic

In this section we focus on the undecidability of the global logic for the global calculus with recursion given in Section 2. In order to prove that the global logic is undecidable, we use a reduction from the Post Correspondence Problem (PCP) [17] similarly to the one proposed in [8]. The idea is to encode in the global calculus a "program" which simulates the construction of PCP. We first give a formal definition of the PCP. In the sequel, $\cdot$ denotes word concatenation.

Definition 4.1 (PCP). *Let $s, t, \ldots$ range over Σ^* where $\Sigma = \{0, 1\}$ and let ε be the empty word. An instance of PCP is a set of pairs of words $\{(s_1, t_1), \ldots, (s_n, t_n)\}$ over $\Sigma^* \times \Sigma^*$. The Post Correspondence Problem is to find a sequence $i_0, i_1, \ldots, i_k$ ($1 \leq i_j \leq n$ for all $0 \leq j \leq k$) such that $s_{i_0} \cdot \ldots \cdot s_{i_k} = t_{i_0} \cdot \ldots \cdot t_{i_k}$.*

Intuitively, PCP consists of finding some string in Σ^* which can be obtained by the concatenation $s_{i_0} \cdot \ldots \cdot s_{i_k}$ as well as by $t_{i_0} \cdot \ldots \cdot t_{i_k}$. Such a problem has been proved to be undecidable [17]. Our goal is to find a GC term that takes a random pair of words from an instance of PCP and append them to an "incremental pair" of words which encodes the current state of the sequences $s_{i_0} \cdot \ldots \cdot s_{i_k}$ and $t_{i_0} \cdot \ldots \cdot t_{i_k}$. Technically, we need a choreography that assigns randomly a natural number in $\{1, \ldots, n\}$ to a variable r in some participant B, and another choreography that picks a pair of words from the PCP instance, accordingly to value in the variable $r@B$, and then appends them to the "incremental pair" of words in A. Formally,

Definition 4.2 (Encoding of PCP). *Let $A_1, \ldots, A_n, A, B$ be participants and a, b shared names for ses-*

sions, then define the two choreographies as shown below:

$$\begin{aligned}
\mathsf{Random}(A_1,\ldots,A_n,B,a) \stackrel{def}{=}\ & \mu X.\, A_1{\to}B{:}a(k).\, A_1{\to}B : k\langle 1,r\rangle.\, X \\
& \mid \mu X.\, A_2{\to}B{:}a(k).\, A_2{\to}B : k\langle 2,r\rangle.\, X \\
& \mid \ldots \\
& \mid \mu X.\, A_n{\to}B{:}a(k).\, A_n{\to}B : k\langle n,r\rangle.\, X
\end{aligned}$$

$$\begin{aligned}
\mathsf{Append}(A,B,b) \stackrel{def}{=}\ & \mu X.\, A{\to}B{:}b(k).\, A{\to}B : k\langle str1,tmp1\rangle.\, A{\to}B : k\langle str2,tmp2\rangle. \\
& \textbf{if } r@B = 1 \textbf{ then} \\
& \quad B{\to}A : k\langle tmp1\cdot s_1, str1\rangle.\, B{\to}A : k\langle tmp2\cdot t_1, str2\rangle.\, X \\
& \textbf{else if } r@B = 2 \textbf{ then} \\
& \quad B{\to}A : k\langle tmp1\cdot s_2, str1\rangle.\, B{\to}A : k\langle tmp2\cdot t_2, str2\rangle.\, X \\
& \textbf{else if } r@B = 3 \textbf{ then} \\
& \quad \vdots \\
& \textbf{else if } r@B = n \textbf{ then} \\
& \quad B{\to}A : k\langle tmp1\cdot s_n, str1\rangle.\, B{\to}A : k\langle tmp2\cdot t_n, str2\rangle.\, X \\
& \textbf{else } X
\end{aligned}$$

We define the initial configuration $(\sigma,\mathscr{C})$ to be formed by the choreography and the state below:

$$\mathscr{C} \stackrel{def}{=} \mathsf{Random}(A_1,\ldots,A_n,B,a) \mid \mathsf{Append}(A,B,b)$$
$$\sigma \stackrel{def}{=} [str1@A \mapsto \varepsilon,\ str2@A \mapsto \varepsilon,\ tmp1@B \mapsto \varepsilon,\ tmp2@B \mapsto \varepsilon,\ r@B \mapsto 1]\,.$$

For encoding the PCP existence question ($s_{i_0}\cdot\ldots\cdot s_{i_k} = t_{i_0}\cdot\ldots\cdot t_{i_k}$) we can encode it as a $\mathscr{GL}$ formula:

$$\phi \stackrel{def}{=} \Diamond\Big((str1@A = str2@A) \wedge (str1@A \neq \varepsilon) \wedge (str2@A \neq \varepsilon)\Big)\,.$$

Above, each participant A_i (with $i \in \{1,\ldots,n\}$) recursively opens a session with participant B and writes in the variable $r@B$ the value i. Moreover, the participant B stores the knowledge of all the word pairs (s_i,t_i), while the participant A takes randomly a word pair from B and then append it to his incremental pair of words: $(str1,str2)$. Next, the formula ϕ states that there exists a computational path from the initial configuration to a configuration which stores in $str1$ and $str2$ two equal non-empty strings.

Theorem 4.3. *The global logic is undecidable on the global calculus with recursion.*

Proof. (Sketch) The statement $\mathscr{C} \models_\sigma \phi$ holds iff the encoded PCP has a solution. Indeed, if the initial configuration $(\sigma,\mathscr{C})$ satisfies the formula ϕ then it means there exists a configuration $(\sigma',\mathscr{C}')$ where $(str1@A = str2@A) \wedge (str1@A \neq \varepsilon) \wedge (str2@A \neq \varepsilon)$ holds. Hence, there is a sequence of $i_0,\ldots,i_k$ such that $str1 = s_{i_0}\cdot\ldots\cdot s_{i_k} = t_{i_0}\cdot\ldots\cdot t_{i_k} = str2$, that is, the instance of PCP has a solution. □

Remark 4.4. The undecidability result presented in this section shows that the global calculus is considerably expressive, despite the choreography approach offers a simplification in the specification of concurrent communicating systems as argued in [5]. The encoding in Definition 4.2 shows that allowing state variables (hence local variables that can be accessed by various threads) increases the expressive power of the language. Indeed, we could just look at GC as a simple concurrent language with a "shared" store where assignment to variables is just in-session communication. In this view, we conjecture that removing variables and focusing only on communication would make the logic decidable.

5 Proof System for Recursion-free Choreographies

In this section, we present a model checking algorithm (in the form of a proof system) to decide whether a global logic formula is satisfied by a recursion-free configuration of the global calculus. Indeed, similarly to [8], it turns out that the logic is decidable on the recursion-free choreographies.[1] We also prove the soundness and completeness of the proposed proof system w.r.t. the assertion semantics.

In order to reason about judgments $\mathscr{C} \models_\sigma \phi$, we propose a proof (or inference) system for assertions of the form $\mathscr{C} \vdash_\sigma \phi$. Intuitively, we want $\mathscr{C} \vdash_\sigma \phi$ to be as approximate as possible to $\mathscr{C} \models_\sigma \phi$ (ideally, they should be equivalent). We write $\mathscr{C} \vdash_\sigma \phi$ for the provability judgement where $(\sigma, \mathscr{C})$ is a configuration and ϕ is a formula.

Notation 5.1. We define the set of continuations configuration after an action ℓ and the reachable configurations, both starting from a configuration $(\sigma, \mathscr{C})$, as follows:

$$\mathsf{Next}(\sigma, \mathscr{C}, \ell) \stackrel{\text{def}}{=} \{(\sigma', \mathscr{C}') \mid (\sigma, \mathscr{C}) \xrightarrow{\ell} (\sigma', \mathscr{C}')\}$$
$$\mathsf{Reachable}(\sigma, \mathscr{C}) \stackrel{\text{def}}{=} \{(\sigma', \mathscr{C}') \mid (\sigma, \mathscr{C}) \longrightarrow^* (\sigma', \mathscr{C}')\}.$$

Normalisation is required by the proof system to infer equality of choreographies up to structural equivalence (Especially for the $[\cdot] \mid [\cdot]$ operator). We define $\mathsf{Norm}(\mathscr{C})$ to be a normalisation function from recursion-free choreographies into multi-sets of choreographies:

$$\mathsf{Norm}(A{\to}B : k\langle e, y\rangle.\ \mathscr{C}) \stackrel{\text{def}}{=} [A{\to}B : k\langle e, y\rangle.\ \mathscr{C}] \qquad \mathsf{Norm}(A{\to}B{:}k[l_i : \mathscr{C}_i]_{i\in I}) \stackrel{\text{def}}{=} [A{\to}B{:}k[l_i : \mathscr{C}_i]_{i\in I}]$$

$$\mathsf{Norm}(A{\to}B{:}a(k).\ \mathscr{C}) \stackrel{\text{def}}{=} [A{\to}B{:}a(k).\ \mathscr{C}] \qquad \mathsf{Norm}(\textbf{if } e@A \textbf{ then } \mathscr{C}_1 \textbf{ else } \mathscr{C}_2) \stackrel{\text{def}}{=} [\textbf{if } e@A \textbf{ then } \mathscr{C}_1 \textbf{ else } \mathscr{C}_2]$$

$$\mathsf{Norm}(\mathbf{0}) \stackrel{\text{def}}{=} [\,] \qquad \mathsf{Norm}(\mathscr{C}_1 \mid \mathscr{C}_2) \stackrel{\text{def}}{=} [P_1, \ldots, P_n, Q_1, \ldots, Q_m] \quad \text{if} \quad \begin{array}{ll} \mathsf{Norm}(\mathscr{C}_1) = [P_1, \ldots, P_n] & \text{and} \\ \mathsf{Norm}(\mathscr{C}_2) = [Q_1, \ldots, Q_m] & . \end{array}$$

Lemma 5.2 (Normalisation preserves structural equivalence). *Let $\mathscr{C}$ be a recursion-free choreography and $\mathsf{Norm}(\mathscr{C}) = [P_1, \ldots, P_n]$, then $\mathscr{C} \equiv \prod_{i=1}^n P_i$.*

Proof. By induction on the structure of the choreography $\mathscr{C}$.

Case $\mathscr{C} = \mathbf{0}$: We have $\mathsf{Norm}(\mathbf{0}) = [\,]$, and $\prod_{i=1}^0 P_i = \mathbf{0} \equiv \mathbf{0}$.

Case $\mathscr{C} = \mathscr{C}_1 \mid \mathscr{C}_2$: We have that $\mathsf{Norm}(\mathscr{C}_1) = [P_1, \ldots, P_n]$, $\mathsf{Norm}(\mathscr{C}_2) = [Q_1, \ldots, Q_m]$, and $\prod_{i=1}^n P_i \equiv \mathscr{C}_1$, $\prod_{j=1}^m Q_j \equiv \mathscr{C}_2$ by induction hypothesis. Then, we can derive that $\prod_{i=1}^n P_i \mid \prod_{j=1}^m Q_j \equiv \mathscr{C}_1 \mid \mathscr{C}_2$.

All the other cases: Trivially we have that $\mathsf{Norm}(\mathscr{C}) = [P_1]$, where $P_1 = \mathscr{C}$, then $\prod_{i=1}^1 P_i \equiv \mathscr{C}$. □

Definition 5.3 (Entailment). *We say that a choreography $\mathscr{C}$* entails *a formula ϕ under a state σ, written $\mathscr{C} \vdash_\sigma \phi$, iff the assertion $\mathscr{C} \vdash_\sigma \phi$ has a proof in the proof system given in Table 4.*

Let us now describe some of the inference rules of the proof system. The rule $\mathsf{P}_{\mathsf{end}}$ relates the inaction terms with the termination formula. The rules $\mathsf{P}_{\mathsf{and}}$ and $\mathsf{P}_{\mathsf{neg}}$ denote rules for conjunction and negation in classical logic, respectively. The rule for parallel composition is represented in $\mathsf{P}_{\mathsf{par}}$; it does not indicate the behaviour of a given choreography, but hints information about the structure of the process: $\mathsf{P}_{\mathsf{par}}$ juxtaposes the behaviour of two processes and combines their respective formulae by the use of a separation operator. The next rule, $\mathsf{P}_{\mathsf{action}}$ requires that the process P in the configuration σ can

[1] Removing recursion yields a decidability result orthogonal to the conjecture formulated in Remark 4.4

$$\mathsf{P}_{\mathsf{end}}\ \frac{\mathsf{Norm}(\mathscr{C}) = [\,]}{\mathscr{C} \vdash_\sigma \mathsf{end}} \qquad \mathsf{P}_{\mathsf{and}}\ \frac{\mathscr{C} \vdash_\sigma \phi \quad \mathscr{C} \vdash_\sigma \chi}{\mathscr{C} \vdash_\sigma \phi \wedge \chi} \qquad \mathsf{P}_{\mathsf{neg}}\ \frac{\mathscr{C} \nvdash_\sigma \phi}{\mathscr{C} \vdash_\sigma \neg\phi}$$

$$\mathsf{P}_{\mathsf{par}}\ \frac{\mathsf{Norm}(\mathscr{C}) = [P_1,\ldots,P_n] \quad \exists I,J.\ I \cup J = \{1,\ldots,n\} \wedge I \cap J = \emptyset \wedge \prod_{i\in I} P_i \vdash_\sigma \phi_1 \wedge \prod_{j\in J} P_j \vdash_\sigma \phi_2}{\mathscr{C} \vdash_\sigma \phi_1 \mid \phi_2}$$

$$\mathsf{P}_{\mathsf{action}}\ \frac{\exists(\sigma',\mathscr{C}') \in \mathsf{Next}(\sigma,\mathscr{C},\ell).\ \mathscr{C}' \vdash_{\sigma'} \phi}{\mathscr{C} \vdash_\sigma \langle\ell\rangle\phi} \qquad \mathsf{P}_{\mathsf{may}}\ \frac{\exists(\sigma',\mathscr{C}') \in \mathsf{Reachable}(\sigma,\mathscr{C}).\ \mathscr{C}' \vdash_{\sigma'} \phi}{\mathscr{C} \vdash_\sigma \Diamond\phi}$$

$$\mathsf{P}_{\exists}\ \frac{\exists w \in fn(\mathscr{C}) \cup fn(\phi).\ \mathscr{C} \vdash_\sigma \phi[w/t]}{\mathscr{C} \vdash_\sigma \exists t.\ \phi} \qquad \mathsf{P}_{\mathsf{exp}}\ \frac{\sigma(e_1@A) \Downarrow v \quad \sigma(e_2@B) \Downarrow v}{\mathscr{C} \vdash_\sigma (e_1@A = e_2@B)}$$

Table 4: Proof system for the Global Calculus.

perform an action labelled ℓ, so we must search for a continuations of $(\sigma,\mathscr{C})$ after an action ℓ and find a configuration which satisfies the rest of the formula, i.e., ϕ. Analogously, $\mathsf{P}_{\mathsf{may}}$ looks for a continuation in the reachable configuration of $(\sigma,\mathscr{C})$ in oder to satisfy ϕ. The rule $\mathsf{P}_{\exists}$ says that in order to satisfy an $\exists t.\ \phi$, it is sufficient to find a value w for t in the free names used by the choreography $\mathscr{C}$ or in the free names used by the formula ϕ. Finally, the rule $\mathsf{P}_{\mathsf{exp}}$ denotes evaluation of expressions.

We now proceed to prove the soundness of the proof system with respect to the semantics of assertions presented before.

Lemma 5.4 (Structural congruence preserves satisfability). *If $\mathscr{C} \equiv \mathscr{C}'$ and $\mathscr{C} \models_\sigma \phi$, then $\mathscr{C}' \models_\sigma \phi$.*

Proof. (Sketch) It follows from structural induction over ϕ. □

Theorem 5.5 (Soundness). *For any configuration $(\sigma,\mathscr{C})$, where $\mathscr{C}$ is recursion-free, and every formula ϕ, if $\mathscr{C} \vdash_\sigma \phi$ then $\mathscr{C} \models_\sigma \phi$.*

Proof. It follows by induction on the derivation of $\vdash_\sigma$.

Case $\mathsf{P}_{\mathsf{end}}$: Straight consequence of Lemmas 5.2 and 5.4, indeed $\mathscr{C} \equiv \mathbf{0}$ and $\mathscr{C} \models_\sigma \mathtt{end}$.

Case $\mathsf{P}_{\mathsf{and}}$: By induction hypothesis and conjunction.

Case $\mathsf{P}_{\mathsf{neg}}$: We have that $\mathscr{C} \vdash_\sigma \neg\phi$, so by $\mathsf{P}_{\mathsf{neg}}$ we get $\mathscr{C} \nvdash_\sigma \phi$. By induction hypothesis we have that $\mathscr{C} \not\models_\sigma \phi$, which is the necessary condition to deduce $\mathscr{C} \models_\sigma \neg\phi$.

Case $\mathsf{P}_{\mathsf{par}}$: We have that $\mathscr{C} \vdash_\sigma \phi_1 \mid \phi_2$, then $\mathsf{Norm}(\mathscr{C}) = [P_1,\ldots,P_n]$, and there exist I,J such that $I \cup J = \{1,\ldots,n\}$, $I \cap J = \emptyset$, $\prod_{i\in I} P_i \vdash_\sigma \phi_1$, and $\prod_{j\in J} P_j \vdash_\sigma \phi_2$. By induction hypothesis we know that $\prod_{i\in I} P_i \models_\sigma \phi_1$ and $\prod_{j\in J} P_j \models_\sigma \phi_2$, then by Lemma 5.2 we have $\mathscr{C} \equiv \prod_{i\in I} P_i \mid \prod_{j\in J} P_j$, hence it is immediate to prove that $\mathscr{C} \models_\sigma \phi_1 \mid \phi_2$.

Case $\mathsf{P}_{\mathsf{action}}$: We have that $\mathscr{C} \vdash_\sigma \langle\ell\rangle\phi$ and by $\mathsf{P}_{\mathsf{action}}$ then $\mathscr{C}' \vdash_{\sigma'} \phi$ and $(\sigma',\mathscr{C}') \in \mathsf{Next}(\sigma,\mathscr{C},\ell)$. From the induction hypothesis we have that $\mathscr{C}' \models_{\sigma'} \phi$, then we have to show that $\mathscr{C} \models_\sigma \rangle\ell\langle\phi$. From the assertion semantics we know that $C \models_\sigma \langle\ell\rangle\phi$ iff $(\sigma,\mathscr{C}') \xrightarrow{\ell} (\sigma',\mathscr{C}')$ and $\mathscr{C}' \models_{\sigma'} \phi$, which holds immediately by the selection of $(\sigma',\mathscr{C}') \in \mathsf{Next}(\sigma,\mathscr{C},\ell)$ and the induction hypothesis.

Case $\mathsf{P}_{\mathsf{may}}$: We have that $\mathscr{C} \vdash_\sigma \Diamond\phi$ and by $\mathsf{P}_{\mathsf{may}}$ then $\mathscr{C}' \vdash_{\sigma'} \phi$ and $(\sigma',\mathscr{C}') \in \mathsf{Reachable}(\sigma,\mathscr{C})$. From the induction hypothesis we have that $\mathscr{C}' \models_{\sigma'} \phi$, then we have to show that $\mathscr{C} \models_\sigma \Diamond\phi$. From the assertion semantics we know that $C \models_\sigma \Diamond\phi \iff (\sigma,\mathscr{C}') \longrightarrow^* (\sigma',\mathscr{C}')$ and $\mathscr{C}' \models_{\sigma'} \phi$, which holds immediately by the selection of $(\sigma',\mathscr{C}') \in \mathsf{Reachable}(\sigma,\mathscr{C})$ and the induction hypothesis.

Case $\mathsf{P}_\exists$**:** We have that $\mathscr{C} \vdash_\sigma \exists t.\phi$ and by $\mathsf{P}_\exists$ we have that $\exists w \in fn(\mathscr{C}) \cup fn(\phi)$ and $\mathscr{C} \vdash_\sigma \phi[w/t]$. By induction hypothesis we know that $C \models_\sigma \phi[w/t]$ with appropriate $w \in fn(\mathscr{C}) \cup fn(\phi)$, then $\mathscr{C} \models_\sigma \exists t.\phi$ follows from the definition of the assertion semantics.

Case $\mathsf{P}_{\mathsf{exp}}$**:** It holds trivially by checking if $\sigma(e_1@A) \Downarrow v$ and $\sigma(e_2@B) \Downarrow v$. □

Lemma 5.6. *For every configuration $(\sigma,\mathscr{C})$, where $\mathscr{C}$ is recursion free, and every formula $\exists t.\ \phi$, if $\{n_1,\ldots,n_k\} = fn(\mathscr{C}) \cup fn(\phi)$, then $\mathscr{C} \models_\sigma \exists t.\ \phi$ iff $\exists m \in \{n_1,\ldots,n_k\}$ such that $\mathscr{C} \models_\sigma \phi[m/t]$.*

Proof. (Sketch) By induction on the structure of ϕ. It is similar to the proof of [7, Lemma 5.3(3)]. □

Theorem 5.7 (Completeness). *For any configuration $(\sigma,\mathscr{C})$, where $\mathscr{C}$ is recursion-free, and every formula ϕ, if $\mathscr{C} \models_\sigma \phi$ then $\mathscr{C} \vdash_\sigma \phi$.*

Proof. By rule induction on the derivation of $\models_\sigma$.

Case $\mathscr{C} \models_\sigma \mathtt{end}$**:** We have that $\mathscr{C} \equiv \mathbf{0}$ and hence $\mathsf{Norm}(\mathscr{C}) = [\,]$ by Lemma 5.2. Now, the thesis follows immediately from the application of $\mathsf{P}_{\mathsf{end}}$.

Case $\mathscr{C} \models_\sigma (e_1@A = e_2@B)$**:** It follows immediately by the application of $\mathsf{P}_{\mathsf{exp}}$.

Case $\mathscr{C} \models_\sigma \langle\ell\rangle\phi'$**:** Take $(\sigma,\mathscr{C}) \xrightarrow{\ell} (\sigma',\mathscr{C}')$ and $\mathscr{C}' \models_{\sigma'} \phi'$, we have by induction hypothesis that $\mathscr{C}' \vdash_{\sigma'} \phi'$. Now, we have to show that $\mathscr{C} \vdash_\sigma \langle\ell\rangle\phi'$. By the fact that $(\sigma,\mathscr{C}) \xrightarrow{\ell} (\sigma',\mathscr{C}')$, we have that $(\sigma',\mathscr{C}') \in \mathsf{Next}(\sigma,\mathscr{C},\ell)$, hence, we can apply rule P_{action} and we are done.

Case $\mathscr{C} \models_\sigma \phi \wedge \chi$**:** We have that $\mathscr{C} \models_\sigma \phi$ and $\mathscr{C} \models_\sigma \chi$. From the induction hypothesis we have that $\mathscr{C} \vdash_\sigma \phi$ and $\mathscr{C} \vdash_\sigma \chi$. The application of $\mathsf{P}_{\mathsf{and}}$ lead to $\mathscr{C} \vdash_\sigma \phi \wedge \chi$ as desired.

Case $\mathscr{C} \models_\sigma \neg\phi$**:** From the definition of the assertion semantics we have that $\mathscr{C} \models_\sigma \neg\phi$ iff $\mathscr{C} \not\models_\sigma \phi$. We have to show that $\mathscr{C} \vdash_\sigma \neg\phi$. We proceed by contradiction. Take a $(\phi,\mathscr{C})$ such that $\mathscr{C} \vdash_\sigma \phi$, then from Theorem 5.5 we have that $\mathscr{C} \models_\sigma \phi$, which is a contradiction to $\mathscr{C} \models_\sigma \neg\phi$.

Case $\mathscr{C} \models_\sigma \exists t.\ \phi$**:** We have that $\mathscr{C} \models_\sigma \exists t.\phi$ and by the definition in the assertion semantics we have that $\mathscr{C} \models_\sigma \phi[w/t]$ for an appropriate w. By induction hypothesis we know that $\mathscr{C} \vdash_\sigma \phi[w/t]$. Lemma 5.6 guarantees that there exists $w \in fn(\mathscr{C}) \cup fn(\phi)$ in order to derive $\mathscr{C} \vdash_\sigma \exists t.\phi$ from $\mathsf{P}_\exists$.

Case $\mathscr{C} \models_\sigma \Diamond\phi$**:** Take $(\sigma,\mathscr{C}) \longrightarrow^* (\sigma',\mathscr{C}')$ and $\mathscr{C}' \models_{\sigma'} \phi'$, we have by induction hypothesis that $\mathscr{C}' \vdash_{\sigma'} \phi'$. Now, we have to show that $\mathscr{C} \vdash_\sigma \Diamond\phi'$. By the fact that $(\sigma,\mathscr{C}) \longrightarrow^* (\sigma',\mathscr{C}')$, we have that $(\sigma',\mathscr{C}') \in \mathsf{Reachable}(\sigma,\mathscr{C})$, hence, we can apply rule P_{may} and we are done.

Case $\mathscr{C} \models_\sigma \phi \mid \chi$**:** We have that $\mathscr{C} \equiv \mathscr{C}_1 \mid \mathscr{C}_2$ and $\mathscr{C}_1 \models_\sigma \phi \wedge \mathscr{C}_2 \models_\sigma \chi$. From the induction hypothesis $\mathscr{C}_1 \vdash_\sigma \phi$ and $\mathscr{C}_2 \vdash_\sigma \chi$. Now by Lemma 5.2 we have that $\mathscr{C}_1 \equiv \prod_{i\in I} P_i$ and $\mathscr{C}_2 \equiv \prod_{j\in J} P_j$ for some I,J. So, we can derive $\mathscr{C} \equiv \prod_{i\in I} P_i \mid \prod_{j\in J} P_j$, and hence $\mathsf{P}_{\mathsf{par}}$ leads to $\mathscr{C}_1 \mid \mathscr{C}_2 \vdash_\sigma \phi \mid \chi$. □

Theorem 5.8 (Termination). *For any configuration $(\sigma,\mathscr{C})$, where $\mathscr{C}$ is recursion-free, and every formula ϕ, proof-checking algorithm terminates.*

Proof. First, notice that all the functions Norm, Next, and Reachable are total and computable. The proof is by induction over the structure of ϕ.

Case $\phi = \mathtt{end}$**:** $\mathscr{C} \vdash_\sigma \mathtt{end}$ iff $\mathsf{Norm}(\mathscr{C}) = [\,]$.

Case $\phi = \phi_1 \wedge \phi_2$**:** By conjunction and induction hypothesis on $\mathscr{C} \vdash_\sigma \phi_1$ and $\mathscr{C} \vdash_\sigma \phi_2$.

Case $\phi = \neg\phi'$: $\mathscr{C} \vdash_\sigma \phi$ iff $\mathscr{C} \vdash_\sigma \phi'$ does not hold. But by induction hypothesis we can construct a terminating proof or confutation for $\mathscr{C} \vdash_\sigma \phi'$. Hence the proof for $\mathscr{C} \vdash_\sigma \phi$ terminates as well.

Case $\phi = \phi_1 \mid \phi_2$: Suppose $\mathsf{Norm}(\mathscr{C}) = [P_1, \ldots, P_n]$. Notice that there exists a finite number of possible partitioning of $\{1, \ldots, n\}$ in I, J. Hence, for every I, J we can compute $\prod_{i \in I} P_i \vdash_\sigma \phi_1$ and $\prod_{j \in J} P_j \vdash_\sigma \phi_2$, which both terminate by induction hypothesis. By applying Lemma 5.2 we prove the thesis.

Case $\phi = \langle \ell \rangle \phi'$: First, notice that the set $\mathsf{Next}(\sigma, \mathscr{C}, \ell)$ is finite, because the choreographies are finite, i.e., there are a finite number of actionable transition in a given configuration. For each configuration $(\sigma', \mathscr{C}') \in \mathsf{Next}(\sigma, \mathscr{C}, \ell)$, $\mathscr{C}' \vdash_{\sigma'} \phi'$ terminates by induction hypothesis.

Case $\phi = \Diamond\phi'$: As before, notice that the set $\mathsf{Reachable}(\sigma, \mathscr{C})$ is finite, because the choreographies are finite, i.e., the choreographies are recursion free. For each configuration $(\sigma', \mathscr{C}') \in \mathsf{Reachable}(\sigma, \mathscr{C})$, $\mathscr{C}' \vdash_{\sigma'} \phi'$ terminates by induction hypothesis.

Case $\phi = \exists t.\ \phi'$: To prove existence is sufficient to check every derivation by substituting t with a name $w \in fn(\mathscr{C}) \cup fn(\phi)$. Notice that $fn(\mathscr{C}) \cup fn(\phi)$ is finite, because both $\mathscr{C}$ and ϕ are so. So, for every w, we can construct a terminating derivation for $\mathscr{C} \vdash_\sigma \phi'[w/t]$ by induction hypothesis.

Case $\phi = (e_1 @ A = e_@ @ B)$: $\mathscr{C} \vdash_\sigma (e_1 @ A = e_@ @ B)$ iff $e_1 @ A \Downarrow v$ and $e_@ @ B \Downarrow v$. □

6 Conclusion and Related Work

The ideas hereby presented constitutes just the first step towards a verification framework for choreography. As a future work, our main concerns relate to integrate our framework into other end-point models and logical frameworks for the specification of sessions. In particular, our next step will focus on relating the logic to the end-point projection [5], the process of automatically generating end-point code from choreography. Other improvements to the system proposed include the use of fixed points, essential for describing state-changing loops, and auxiliary axioms describing structural properties of a choreography.

This work can be fruitfully nourished by related work in types and logics for session-based communication. In [13] the authors proposed a mapping between the calculus of structured communications and concurrent constraint programming, allowing them to establish a logical view of session-based communication and formulae in First-Order Temporal Logic. In [1], Berger et al. presented proof systems characterising May/Must testing pre-orders and bisimilarities over typed π-calculus processes. The connection between types and logics in such system comes in handy to restrict the shape of the processes one might be interested, allowing us to consider such work as a suitable proof system for the calculus of end points. Finally, [15] studies a logic for choreographies in a model without services and sessions while [2] proposes notion of global assertion for enriching multiparty session types with simple formula describing changing in the state of a session.

Acknowledgements This research has been partially supported by the Trustworthy Pervasive Healthcare Services (TrustCare) and the Computer Supported Mobile Adaptive Business Processes (Cosmobiz) projects. Danish Research Agency, Grants # 2106-07-0019 (www.TrustCare.eu) and # 274-06-0415 (www.cosmobiz.org).

References

[1] Martin Berger, Kohei Honda & Nobuko Yoshida (2008): *Completeness and Logical Full Abstraction in Modal Logics for Typed Mobile Processes*. In Luca Aceto, editor: *ICALP'08*, *LNCS* 5126, Springer-Verlag, Berlin Germany, pp. 99–111, doi:10.1007/978-3-540-70583-3_9.

[2] Laura Bocchi, Kohei Honda, Emilio Tuosto & Nobuko Yoshida (2010): *A theory of design-by-contract for distributed multiparty interactions*. In: *CONCUR'10: Proceedings of the 21st International Conference on Concurrency Theory*, Lecture Notes in Computer Science, Springer - Verlag, pp. 162–176, doi:10.1007/978-3-642-15375-4_12.

[3] L. Caires & L. Cardelli (2001): *A spatial logic for concurrency (part I)*. In: *Theoretical Aspects of Computer Software*, Springer, pp. 1–37, doi:10.1007/3-540-45500-0_1.

[4] M. Carbone, K. Honda & N. Yoshida (2007): *A Calculus of Global Interaction based on Session Types*. In: *2nd Workshop on Developments in Computational Models (DCM)*, ENTCS, pp. 127–151, doi:10.1016/j.entcs.2006.12.041.

[5] M. Carbone, K. Honda & N. Yoshida (2007): *Structured communication-centred programming for web services*. In: *16th European Symposium on Programming (ESOP)*, *LNCS* 4421, Springer, Berlin Heidelberg, Braga, Portugal, pp. 2–17, doi:10.1007/978-3-540-71316-6_2.

[6] M. Carbone, K. Honda, N. Yoshida, R. Milner, G. Brown & S. Ross-Talbot (2009): *A Theoretical Basis of Communication-Centred Concurrent Programming*. *Web Services Choreography Working Group mailing list, WS-CDL working report* .

[7] Luca Cardelli & Andrew D. Gordon (2000): *Anytime, Anywhere: Modal Logics for Mobile Ambients*. In: *POPL*, pp. 365–377, doi:10.1145/325694.325742.

[8] Witold Charatonik & Jean-Marc Talbot (2001): *The Decidability of Model Checking Mobile Ambients*. In Laurent Fribourg, editor: *CSL*, *Lecture Notes in Computer Science* 2142, Springer, pp. 339–354, doi:10.1007/3-540-44802-0_24.

[9] E.A. Emerson (1991): *Temporal and modal logic*. In: *Handbook of theoretical computer science (vol. B)*, MIT Press, p. 1072.

[10] M. Hennessy & R. Milner (1980): *On Observing Nondeterminism and Concurrency*. In: *Proceedings of the 7th Colloquium on Automata, Languages and Programming*, Springer-Verlag London, UK, pp. 299–309, doi:10.1007/3-540-10003-2_79.

[11] K. Honda, V.T. Vasconcelos & M. Kubo (1998): *Language Primitives and Type Discipline for Structured Communication-Based Programming*. In: *7th European Symposium on Programming (ESOP): Programming Languages and Systems*, Springer-Verlag London, UK, pp. 122–138, doi:10.1007/BFb0053567.

[12] N. Kavantzas, D. Burdett, G. Ritzinger, T. Fletcher, Y. Lafon & C. Barreto (2004): *Web services choreography description language version 1.0*. *W3C Working Draft* 17, pp. 10–20041217.

[13] Hugo A. López, Carlos Olarte & Jorge A. Pérez (2010): *Towards a Unified Framework for Declarative Structured Communications*. In: *Programming Language Approaches to Concurrency and Communication-cEntric Software (PLACES'2009)*, *EPTCS* 17, pp. 1–15, doi:10.4204/EPTCS.17.1.

[14] Robin Milner (1999): *Communicating and Mobile systems. The Pi Calculus*. Cambridge University Press.

[15] Carlo Montangero & Laura Semini (2006): *A Logical View of Choreography*. In: *COORDINATION*, pp. 179–193, doi:10.1007/11767954_12.

[16] G. D. Plotkin (1981): *A Structural Approach to Operational Semantics*. Technical Report, University of Aarhus.

[17] Emil L. Post (1944): *Recursively enumerable sets of positive integers and their decision problems*. *Bulletin of the American Mathematical Society* 50, pp. 284–316.

[18] JC Reynolds (2002): *Separation logic: a logic for shared mutable data structures*. *Logic in Computer Science, 2002. Proceedings. 17th Annual IEEE Symposium on* , pp. 55–74Available at `http://doi.ieeecomputersociety.org/10.1109/LICS.2002.1029817`.