ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN

COMPUTER SCIENCE AND ENGINEERING

Ciclo 38

Settore Concorsuale: 01/B1 - INFORMATICA

Settore Scientifico Disciplinare: INF/01 - INFORMATICA

A CHOREOGRAPHIC APPROACH TO MODELLING AND ANALYSIS OF ADMINISTRATIVE PROCEDURES IN HEALTHCARE MANAGEMENT

Presentata da: Sourabh Pal

Coordinatore Dottorato

Prof. Paola Salomoni

Supervisore

Prof. Ivan Lanese

Co-supervisore

Luca Cisbani

Esame finale anno 2026

DECLARATION

I hereby declare that the work presented in this dissertation entitled ”A Choreographic Approach to Modelling and Analysis of Administrative Procedures in Healthcare Management” is the result of my own original research carried out under the supervision of Prof. Ivan Lanese at Department of Computer Science and Engineering of University of Bologna, Italy.

This dissertation has not been submitted, either in whole or in part, for the award of any other degree or diploma at this or any other university. All sources of information and data used in this work have been duly acknowledged.

This thesis has been Funded by the European Union - Next Generation EU, Mission 4, Component 2, Investment 3.3 (Ministerial Decree 117/2023) CUP E53C24001950001.

Place: Bologna
Date: 11/02/2026

Sourabh Pal

Finanziato dall'Unione europea NextGenerationEU | Ministero dell'Università e della Ricerca | Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA | ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

ACKNOWLEDGMENT

I would like to express my deepest gratitude to my supervisor, Prof. Ivan Lanese, for his invaluable guidance, continuous support, and insightful feedback throughout my PhD journey. My sincere thanks go to Luca Cisbani, my co-supervisor from public administration at Salute e Welfare - Regione Emilia Romagna, to provide the case study for my research and feedback about the results that have enriched this work. I am also deeply grateful to my external supervisor and collaborator from KTH, Prof. Roberto Guanciale, for his thoughtful advice for the extension of the tool PomCho. I would also like to acknowledge Prof. Emilio Tuosto for his valuable collaboration during the work on g-choreographies and the extension of the PomCho tool.

My heartfelt thanks extend to my Ph.D coordinator Prof. Ilaria Bartolini, whose administrative support has been essential in navigating the various stages of this doctoral program. I am thankful to all my colleagues for their stimulating discussions, and for creating a supportive and inspiring research environment.

Finally, I owe my deepest gratitude to my parents, whose unwavering love, sacrifices, and constant encouragement have been my greatest source of strength and motivation throughout this journey.

Abstract

Choreographic models in general, and *Choreographic Automata* (CA) and *global choreographies* (*g-choreography*) in particular, can be used to analyse and validate communicating systems. This thesis applied these models to a case study in healthcare management, the procedure for *accreditation and authorisation* of public and private healthcare structures in the Emilia Romagna region (Italy). Complex coordination protocols are necessary to manage complex organisations. The healthcare management sector is no exception, since different authorities, users, and systems have to interact with each other in order to achieve their organisational goals. In this thesis, we verified the *realisability* of communicating systems using CA and *g-choreography* in general, and of the case study in particular. The main contributions of this thesis are as follows:

(i) The formalisation of the case study via BPMN and choreographic models;

(ii) The analysis of the protocol using Corinne and PomCho;

(iii) The theoretical and algorithmic extensions introduced for PomCho to check *realisability*;

(iv) The lessons learned from applying these techniques to a real administrative case study;

The case study was originally described in natural language. Therefore, we first formalised the procedure using a BPMN collaboration diagram as an intermediate step. Initially, we converted the BPMN model into CAs. Afterward, we

verify the *correctness* of the system using the Corinne tool, as its underlying theory is CAs. The tool Corinne showed a few issues in the formalised model, but it turned out that such issues were due to too strict requirements posed by the theory underlying Corinne. This gave us useful feedback for future improvements of Corinne and its underlying theory.

Due to some limitation in the CA and Corinne such as the explosion of the size of the model due to the fact that CAs model *concurrency* by explicitly representing all the *interleavings*, we moved to *g-choreographies* so to enable the analysis of the *correctness* of its communication patterns using the PomCho tool. This requires to refine PomCho and its underlying theoretical framework. First, we extend PomCho to support not only *asynchronous* communication, but also *synchronous* one. Moreover, in both the cases, we provide *a more efficient algorithm* to check *closure* properties ensuring realisability of choreographies. We refer to the extended version of PomCho as PomCho+. The new algorithm allows us to check realisability of larger pomsets than before, which makes our approach viable for complex systems such as our case study.

Keywords: Choreographies; Healthcare management; Modelling; Pomsets; Communicating systems.

Table of Contents

Lists of Figures **iii**

1 Introduction **1**
1.1 Protocol Formalisation and Analysis 1
1.2 Contributions and Structure of the Thesis 4
1.3 Publications 5

2 Background **6**
2.1 A Bird-Eye View of Choreographies 6
2.1.1 Choreography Automata 8
2.1.2 G-Choreographies 13
2.2 Asynchronous Semantics 15

3 Case Study **21**
3.1 A&A: Natural Language Description 21
3.2 A&A: BPMN Collaboration Diagram 25

4 Analysis with Corinne **30**
4.1 A&A: Choreographic Automaton 30
4.2 Validation 34
4.3 Lessons Learned 36
4.4 Publication 39

5 Analysis with PomCho **40**
5.1 A&A: Process Management in Italian Healthcare 40
5.2 Synchronous Semantics 45
5.3 Verifying Realisability 49
5.3.1 Avoiding pomset compositions explosion 49
5.3.2 Avoiding prefix explosion 52
5.4 Evaluation 54
5.4.1 Verifying the closure conditions on A&A 55
5.4.2 Benchmarking our algorithms 58
5.5 Publication 61

6 Related Work **62**

7 Conclusion and Future Work **66**

References **70**

List of Figures

2.1 An example of CA 9
2.2 Example of *projection*. 10
2.3 Example of *well-sequencedness*. 11
2.4 Runs of CA Figure 2.1 12
2.5 Diagrammatic Representation of the Syntax of Global Choreographies 14

3.1 BPMN Collaboration Diagram: A Fragment of A&A. 27

4.1 Choreographic Automaton for Regional Coordination for Authorisation and Accreditation. 31

5.1 G-choreography for Regional Coordination for Authorisation and Accreditation 42
5.2 Realisable G-choreography of the A&A 56
5.3 G_{xy} with $n > 4$ and $n < 46$ with respect to Time (s) 60

Chapter 1

Introduction

1.1 Protocol Formalisation and Analysis

Distributed software systems present a fundamental tension: while the decomposition of functionality across multiple components helps managing complexity and improve efficiency, it also scatters execution contexts and states across the computational environment. Therefore, it becomes difficult to guarantee that global properties hold. Centralised coordination can enforce those properties but introduces performance bottlenecks and single points of failure. Consequently, distributed components must exchange exactly the right information so that each local state contributes correctly to a globally consistent behavior, an inherently challenging task.

Due to the challenges in contemporary distributed systems, choreographic models (see, e.g., the survey in [1]) are gaining popularity in the design and implementation of distributed systems in different domains [2] e.g., healthcare systems [3]. This is demonstrated by the creation of standards like BPMN [4], as well as by their uptake for contemporary architectures like microservices [5]. Their application in *message-passing system* modelling, analysis, and programming has been studied. This includes syntax-free approaches [6], formalisation through behavioral types [7], analysis of liveness and deadlock freedom [1,8,9], and verification of BPMN specifications [10]. More specifically, the choreographic approaches are used to holistically describe the interactions among components in *message-passing systems*, and enable to enforce relevant communication properties such as *deadlock-freedom* by construction. Choreographic programming has also been extensively researched at the programming level [11, 12].

Choreographies provide two complementary views, i.e., the *global view* and the

local view. The *global view* [1] is the abstract description of the entire system, i.e., it consists of all the participants and describes their interactions from a global viewpoint. This global perspective aims to simplify the modelling of interactions and their ordering constraints. On the contrary, the *local view* represents the behavior of a specific participant and the message exchanges involving this participant. The expected behaviour of each component can be captured by formal models such as *Communicating Finite State Machines* [13] or programming languages featuring message-passing (e.g., Erlang or GoLang). The *global view* is useful to design a system, while *local views* are more relevant in the implementation phase. Given a *global view*, it is possible to automatically generate *local views* enacting the global behavior. Unfortunately, not every choreography admits a correct realisation in a set of local behaviours. To bridge this gap, we have explored two approaches, i.e., Choreographic Automata (CAs) [14] and *g-choreographies* [15], for analysing and verifying the *correctness* of the system, so that distributed systems can be implementable.

Let us first discuss CAs. They combine the choreographic approach with the visual representation and the mathematical theory of *finite state automata*. We apply CAs to a case study from healthcare administration, namely the procedure for accreditation and authorisation of public and private healthcare structures in the Emilia Romagna region (Italy). We refer to this case study as A&A, denoting *A*ccreditation *and A*uthorisation, throughout the thesis. Since A&A is defined in natural language, in form of a regional law, formalising it directly as CA is not easy. We found a suitable intermediate step as a collaboration diagram in the Business Process Modelling Notion (BPMN) [16]. BPMN is a semi-formal graphical notation for describing business processes, and it is more flexible than CAs. Such flexibility allows for easier modelling and easier interaction with domain experts. Then, one can refine the BPMN collaboration diagram to derive a CA, more suitable for the automatic analysis of communication properties.

For the analysis, we exploit the tool Corinne [17,18], which allows one to analyse CAs represented in the DOT format [19]. On the one hand, this allows us to verify that the protocol considered in the A&A satisfies relevant communication properties. On the other hand, this allows us to verify also whether the expressive power of CAs and the related analysis techniques are suitable for applications in such a setting. Although CAs inherently support concurrency, their underlying representation and theoretical foundations are highly complex. Consequently, the modelling of concurrency in CAs leads to an increased number of interleavings, which, in turn, amplifies the complexity of verifying system

correctness. Actually, our analysis of the A&A showed that the theory implemented in Corinne (and aligned with most of the theories for choreographies in the literature, such as the ones surveyed in [1]) is too rigid, signalling as errors behaviors that are not problematic in practice. These issues primarily stem from the limited handling of concurrency and the lack of support for *late join* (also called selective participation) [20]. *Late join*, it requires that if a participant is involved in some branch of a protocol, then it should be involved in all the branches. The rationale for this is that if a branch in which the participant is not involved is taken, then the participant is left waiting, possibly forever. However, in practice, participants are not waiting for interaction, but just react if they receive a request.

In this thesis, to bridge this gap, we address two related challenges, namely late join and avoiding explicit interleavings. To do so, we provide a language-independent solution based on some realisability conditions on the events in the *global view* of the *g-choreographies*. Concretely, we employ *partially ordered multisets (pomsets)* as an abstract representation of global specifications. A pomset exposes the causal relationships among send and receive events without prescribing a total order, making it equally suitable for reasoning about *synchronous* rendezvous and *asynchronous* semantics. By analysing only the partial order of events, we formulate syntax-independent realisability conditions that can be checked directly on pomsets—eliminating the need to enumerate interleavings or translate into a particular choreography language. Hence, the *g-choreographies* resolve the issues of interleavings. By operating directly on pomsets, we enable designers to detect and eliminate coordination mismatches at design time, rather than discovering them as costly implementation bugs.

In analysing the A&A, we employed the PomCho [21] tool and developed its extended version, PomCho+ [22], which supports the analysis of *g-choreographies* represented in the SGG format [23]. The analysis of the A&A was performed using PomCho+. Initially, PomCho was implemented in a rather naive manner, which limited its efficiency when handling large-scale case studies involving a substantial number of interleavings. To overcome this limitation, we introduced a novel algorithm in PomCho+, making the tool computationally efficient in terms of both time and space complexity. Moreover, the original version of PomCho was capable of analysing only asynchronous semantics. To enhance its analytical capabilities, we extended the tool to support synchronous semantics as well. Consequently, PomCho+ can now analyse both asynchronous and synchronous semantics. Upon reanalysing the A&A using PomCho+, we observed that all the issues previously reported in Corinne, particularly those

associated with late join, were resolved. However, one common error was identified in both the CA and *g-choreography* analyses. Notably, this error did not stem from the protocol itself but rather from inaccuracies introduced during the system modelling process.

1.2 Contributions and Structure of the Thesis

In this thesis, we analyse the A&A using CA and *g-choreographies* to verify realisability. To this end, we use the existing tools Corinne and PomCho. The theoretical foundation of Corinne is based on CA, whereas PomCho is grounded in the theory of *g-choreographies*. We have analysed A&A using Corinne. The A&A consists of multiple concurrent protocols. Due to some limitation in the CA and Corinne such as the explosion of the size of the model due to the fact that CAs model concurrency by explicitly representing all the interleavings, we moved to *g-choreographies* so to enable the analysis of the correctness of its communication patterns using the PomCho tool. However, the existing version of PomCho supported realisability checking only in the *asynchronous* setting. Moreover, its underlying algorithm did not scale sufficiently to verify large systems such as our A&A. In this thesis, we address these limitations by proposing a new algorithm capable of analysing significantly larger models. In addition, we extend PomCho to support *synchronous* communication. Finally, we derive lessons learned from the analysis using both approaches. This dissertation contains the following chapters:

- **Chapter 2**: It provides the background of CAs and the properties of CAs that need to be checked to verify the correctness of the system. It also introduces the concept of *g-choreographies* and the *asynchronous* model, along with its *pomset* semantics, to verify the realisability of the system and determine whether it is implementable.
- **Chapter 3**: It provides an informal introduction to the A&A in Section 3.1. The A&A is then modelled as a BPMN collaboration diagram in Section 3.2.
- **Chapter 4**: It provides the CA based model of A&A in Section 4.1. We analyse the CA using Corinne in Section 4.2 and we discuss the lessons learned from the modelling and validation in Section 4.3.
- **Chapter 5**: It provides the *g-choreography* based model of A&A in Sec-

tion 5.1 whose size challenges both the theory [24] and the implementation in [23]. We refine the pomset framework to encompass synchronous communication, defining new realisability conditions for rendezvous-style send/receive primitives in Section 5.2. We introduce the algorithm of *asynchronous* and *synchronous* model for verification of realisability over large pomsets in Section 5.3. It provides the evaluation of the A&A and the benchmark of the proposed algorithm with respect of the previous one in Section 5.4.

- **Chapter 6**: It presents the related work.
- **Chapter 7**: It describes the conclusion and outlines directions for future work.

1.3 Publications

The material in this thesis has been published in two conference papers. The papers are as follows:

- **Sourabh Pal**, Roberto Guanciale, Ivan Lanese, Emilio Tuosto, and Massimo Clo. ”Pomsets for Process Management: A Healthcare Case Study.” In International Colloquium on Theoretical Aspects of Computing, pp. 378-395. Cham: Springer Nature Switzerland, 2025.
- **Sourabh Pal**, Ivan Lanese, and Massimo Clo. ”Choreographic Automata: A Case Study in Healthcare Management.” International Conference on Coordination Models and Languages. Cham: Springer Nature Switzerland, 2024.

Chapter 2

Background

2.1 A Bird-Eye View of Choreographies

Choreographies allow one to describe distributed systems composed of multiple participants which interact via message-passing, like formalisms such as *Message Sequence Charts* (MSC) [25, 26], multiparty session types [7], and BPMN choreographies [27]. Following the choreographic approach to system design (see, e.g., [1]), distributed systems can be represented using two complementary views, namely a single *global view* or multiple *local views*, one per participant. The *global view* describes the order in which messages can be exchanged inside a system from a global point of view. A single *global view* corresponds to multiple *local views*, each of which defines the behaviour of a participant. Broadly speaking, the *global view* is more suitable for specification and understanding, while the *local view* is closer to the final implementation. Both views focus on communication, abstracting away computation.

An interaction $\mathsf{A} \to \mathsf{B} : \mathsf{m}$ denotes that, participant A sends message m to participant B. A interaction represents the combination of an *output* event and an *input* event. An *output* event, also called a sending action, occurs when a participant sends a message to another participant, whereas an *input* event, also called a receiving action, occurs when a participant receives a message from

another participant. Let $\mathcal{P}$ be a set of *participants*, $\mathcal{M}$ a set (of types) of *messages*. We take $\mathcal{P}$ and $\mathcal{M}$ disjoint. Participants coordinate with each other by exchanging messages over *communication channels*, that are elements of the set $\mathcal{C} = (\mathcal{P} \times \mathcal{P}) \setminus \{(\mathsf{A},\mathsf{A}) \mid \mathsf{A} \in \mathcal{P}\}$. We abbreviate $(\mathsf{A},\mathsf{B}) \in \mathcal{C}$ as $\mathsf{A}\cdot\mathsf{B}$. The set of *(communication) labels* $\mathcal{L}$ is defined by

$$\mathcal{L} = \mathcal{L}^! \cup \mathcal{L}^? \qquad \text{where} \qquad \mathcal{L}^! = \mathcal{C} \times \{!\} \times \mathcal{M} \qquad \text{and} \qquad \mathcal{L}^? = \mathcal{C} \times \{?\} \times \mathcal{M}$$

The elements of $\mathcal{L}^!$ and $\mathcal{L}^?$, outputs and inputs, respectively represent *sending* and *receiving* actions. $\mathsf{A}\cdot\mathsf{B}!\mathsf{m}$ $(\mathsf{A}\cdot\mathsf{B},!,\mathsf{m})$ is a sending action, where as $\mathsf{A}\cdot\mathsf{B}?\mathsf{m}$ $(\mathsf{A}\cdot\mathsf{B},?,\mathsf{m})$ is a receiving event.

Projection is the process of deriving the local behaviour of each participant from a global specification of a system's interactions. A choreography describes the overall communication flow among participants—while a projection represents each participant's viewpoint. *Correctness* or *realisability* refers to whether a global specification can be faithfully implemented by a collection of local specifications. A choreography describes the intended overall interaction between participants, but when it is projected into local behaviours, those components may not always coordinate exactly as intended due to issues like message reordering, concurrency, or missing synchronisation. Correctness ensures that, when all local processes execute and exchange messages, their combined behaviour matches exactly the behaviour described by the global specification. In other words, a global specification is realisable if it is implementable without introducing unintended behaviours. In this case, properties such as the absence of deadlocks and message mismatches typically follow. In short, projection bridges the gap between global specifications and implementable local processes. We can verify the implementability of a system in two different ways, namely using CAs and *g-choreographies*, as discussed below:

2.1.1 Choreography Automata

CAs are an expressive and flexible model of global specifications, following the styles of conversation protocols [28], choreographies [4, 29], global graphs [6] and multiparty session types [1, 7, 30].

A CA is a finite-state model that describes the global behaviour of a distributed system in terms of the interactions among its participants. Unlike process-based models, which define the behaviour of each participant individually, a CA specifies the overall communication protocol as a single automaton whose transitions represent message exchanges between roles.

Definition 1 (Choreography Automaton)**.** *A CA is a tuple*

$$\mathcal{A} = \langle Q, q_0, \Sigma, \delta \rangle$$

where:

- Q *is a finite set of* global states*;*
- $q_0 \in Q$ *is the* initial state*;*
- Σ *is a finite* alphabet of global actions*; and*
- $\delta \subseteq Q \times \Sigma \times Q$ *is the* transition relation*.*

Each global action $\sigma \in \Sigma$ *is of the form*

$$\mathsf{A} \rightarrow \mathsf{B} : \mathsf{m}$$

where A *and* B *are participants (or roles) and* m *is a message label.*

Example 2.1.1. *Consider the example of a CA shown in Figure 2.1. There are three participants in this scenario, namely: the Customer (*C*), the Bank (*B*), and the Fraud*

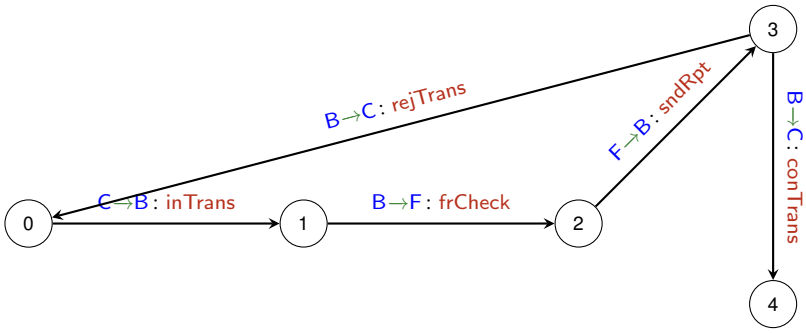


Figure 2.1: An example of CA

Detector (F). First, the customer initiates the transfer by sending a message to the bank: C→B : inTrans*, from state* 0 *to state* 1*. Afterwards, the bank requests the fraud detector to check whether the transfer is fraudulent by sending the message* B→F : frCheck *from state* 1 *to state* 2*. The fraud detector then sends a report about the transfer with the message* F→B : sndRpt *from state* 2 *to state* 3*. If the transfer is not fraudulent, the bank confirms the transfer with the message* B→C : conTrans *from state* 3 *to state* 4*; otherwise, if it is fraudulent, the bank rejects the transfer with the message* B→C : rejTrans *from state* 4 *back to state* 0*.*

From a CA, the local behaviour of each participant can be obtained by a projection operation. The projection for a participant A yields a finite-state automaton that includes a send action $\mathsf{A} \cdot \mathsf{B}!\mathsf{m}$ whenever A acts as sender in $\mathsf{A} \to \mathsf{B} : \mathsf{m}$, and a receive action $\mathsf{A} \cdot \mathsf{B}?\mathsf{m}$ whenever B is the receiver.

Definition 2 (Projection [14])**.** *The projection on participant* $\mathsf{A} \in \mathcal{P}$ *of a transition*

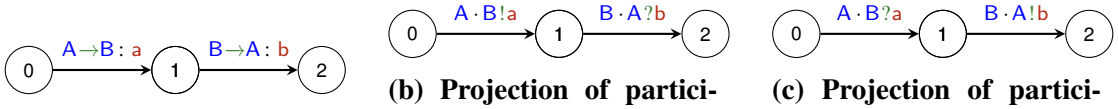
$$t = q \xrightarrow{\lambda} q'$$

of a choreography*, written* $r\!\upharpoonright_{\mathsf{A}}$*, is defined by:*

$$r\!\upharpoonright_{\mathsf{A}} = \begin{cases} q \xrightarrow{\mathsf{A}\cdot\mathsf{B}!\mathsf{m}} q' & \text{if } \lambda = \mathsf{A} \to \mathsf{B} : \mathsf{m}, \mathit{OR} \\ q \xrightarrow{\mathsf{A}\cdot\mathsf{B}?\mathsf{m}} q' & \text{if } \lambda = \mathsf{A} \to \mathsf{B} : \mathsf{m}, \\ q \xrightarrow{\varepsilon} q' & \text{otherwise.} \end{cases}$$

The projection function trivially extends to choreography words and languages. The projection defined above, apart for determinisation, is essentially homomorphic, as most of the projections in the literature. Other approaches such as [31, 32] add hidden communications to be able to deal with larger classes of choreographies. We prefer the former approach for its simplicity. Hidden communications can however be added directly at the choreographic level as proposed in [33].

Example 2.1.2. *Consider the CA in the Figure 2.2a, and it's corresponding projection with respect to participant A in Figure 2.2b and B in Figure 2.2c. Although the diagrammatic representation of the* g-choreography *differs from that of the CA, their projections are identical.*



(a) Example of CA

(b) Projection of participant A

(c) Projection of participant B

Figure 2.2: Example of *projection*.

To ensure that the projected local automata, when composed, faithfully implement the intended global behaviour, the original CA must satisfy two structural properties: *well-sequencedness*, ensuring that causally independent interactions do not interfere, and *well-branchedness*, ensuring that alternative behaviours are governed by a single participant (the branch selector). When both conditions hold, the CA is said to be *well-formed*. For well-formed choreographies, the composition of their projections is guaranteed to be *live*, *deadlock-free*, and *lock-free* under both synchronous and asynchronous semantics. In this discussion, we will restrict our attention to synchronous semantics.

The following definitions are taken from [14].

Definition 3 (Well-sequencedness)**.** *A c-automaton CA is well-sequenced if for each two consecutive transitions* $q \xrightarrow{A \to B : m} q' \xrightarrow{C \to D : n} q''$ *either*

- *They share a participant, that is* $\{\mathsf{A},\mathsf{B}\} \cap \{\mathsf{C},\mathsf{D}\} \neq \emptyset$*, or*
- *They are concurrent, i.e. there is* q''' *such that* $q \xrightarrow{\mathsf{C}\to\mathsf{D}:\mathsf{n}} q''' \xrightarrow{\mathsf{A}\to\mathsf{B}:\mathsf{m}} q''$

Intuitively, two transitions that share no participant are independent, hence can be executed in any order, and well-sequencedness requires that both orders are explicitly represented.

Figure 2.3: Example of *well-sequencedness*.

Example 2.1.3. *Not all CAs are* well-sequenced*. Consider the CA in Figure 2.3a, which is not* well-sequenced *because of the transitions from state* 0 *to state* 1 *and from state* 1 *to state* 2*, as there is no common participant between these two transitions. By "completing the diamond" for such transitions (i.e., by adding the new state* 3 *and the corresponding transitions), we obtain the CA shown in Figure 2.3b. Now, it is* well-sequenced*.*

In order to define well-branchedness we need some preliminary definitions.

Definition 4 (Candidate q-branch)**.** *A pre-candidate q-branch is a path in the automaton starting from state q and such that each cycle has at most one occurrence within the path. A candidate q-branch is a maximal pre-candidate q-branch with respect to prefix order.*

Example 2.1.4. *The sequence in Figure 2.4 are runs of the CA in Figure 2.1. They are all pre-candidates of either state* 2 *or* 3*, but the run* π_c *is not a pre-candidate of state* 2 *because of the cycle* $2-3-0-1-2$*,* $3-0-1-2-3$*, and* $0-1-2-3-0$*.*

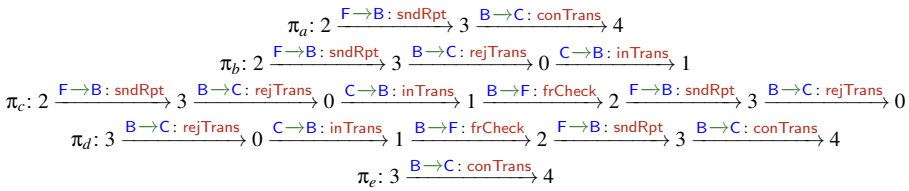


Figure 2.4: Runs of CA Figure 2.1

Definition 5 (q-span)**.** *A pair (π,π') of pre-candidate q-branches of CA is a q-span if*

1. *either π and π' are cofinal, with no common node but q and the last one;*
2. *or π and π' are candidate q-branches with no common node but q;*
3. *or π and π' are a candidate q-branch and a loop on q with no other common nodes.*

Example 2.1.5. *The states with spans in our example (Figure 2.1) are* 2 *and* 3*. State* 2 *has no spans, since the first transition is unique. A span from state* 3 *is the pair (π_d,π_e), where π_d and π_e are the runs shown in Figure 2.4. The runs π_d and π_e are cofinal in state* 4 *and are pre-candidates of state* 3*, sharing no common states except the initial and final ones, i.e., states* 3 *and* 4*. But, π_d can be stopped at* 3*, fitting item* 3 *of Definition 5. In contrast, the pair (π_b,π_c) does not form a span because the runs have different cofinal states and share intermediate states other than those that are pre-candidates of state* 2*.*

We can now define well-branchedness. Well-branchedness is defined only on deterministic CAs, but A&A is deterministic hence this restriction has no impact.

Definition 6 (Well-branchedness)**.** *A deterministic CA is well-branched if for each state q and participant A sender in a transition from q, all of the following conditions hold:*

1. *all transitions from q involving A, have sender A;*

2. *for each transition t from q whose sender is not A and each transition t' from q whose sender is A, t and t' are concurrent;*

3. *for each q-span (π, π') where A is the sender of the first transition in both paths and each participant $\mathsf{B} \neq \mathsf{A}$, consider the first pair of different transitions involving B in π and in π': they are both receive, with a different sender or a different message.*

Intuitively, well-branchedness ensures that when a participant A makes a choice between two branches π and π', it makes the choice between all the available options (condition 1) unless they are concurrent actions (condition 2). Also, each other participant B, if it has to make different transitions in the two branches, it is notified of the outcome of the choice by either receiving different messages, or receiving messages from different participants (condition 3).

Such conditions should hold on any pair of paths starting with a common node, but actually it is enough to make the check only on maximal pairs, which form what we call q-spans.

2.1.2 G-Choreographies

In order to overcome some limitations of CAs, we also used *g-choreographies* (*global choreographies*, in the terminology of the literature) [6] to represent global views, and *Communicating Finite State Machines* [13] to represent local views. A g-choreography is a term derivable from the following grammar:

$$\mathsf{G} \;::=\; \mathsf{A}{\to}\mathsf{B}:\mathsf{m} \;\mid\; \mathsf{G};\mathsf{G}' \;\mid\; \mathsf{G} \mid \mathsf{G}' \;\mid\; \mathsf{G} + \mathsf{G}' \;\mid\; \mathit{repeat}\ \mathsf{G}$$

G-choreographies can be composed in sequence, in parallel, in nondeterministic

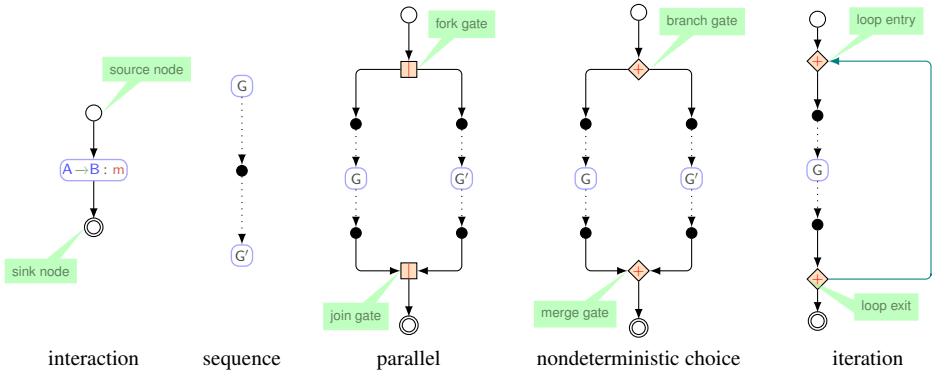


Figure 2.5: Diagrammatic Representation of the Syntax of Global Choreographies

choice, or iterated. Each g-choreography in the syntax above has a graphical representation that can be formally derived by induction on the syntax of the g-choreography. Instead of a formal definition, we show how to achieve this using the sketches in Fig. 2.5. Intuitively, G can be depicted as a graph rooted in a *source* node $\circ$ and having a single *sink* node $\circledcirc$. Connecting nodes represent interaction, branch or fork points, or else looping points; this is immediate in the representation of interactions in Fig. 2.5. Besides interaction nodes, our graphical notation uses *gate* nodes $\square$ and $\diamond$ to identify fork and branch or iteration as well as their corresponding "closing" points. Gate nodes and dotted edges allow us to compose the diagrams; more precisely, a dotted edge from $\bullet$ to a boxed G (the diagrammatic representation of G) means that the gate connected to the source node of G should be connected to the gate entering in $\bullet$ and that the source node of G is removed; similarly, a dotted edge from a boxed G to $\bullet$ means that the gate arriving in the sink node of G has to be connected to the gate after $\bullet$ and that the sink node of G is removed. For instance, in the graph for the sequential composition, the top-most edge identifies the sink node of G and the other edge identifies the source node of G'.

We will describe the asynchronous and synchronous semantics of choreographies in terms of pomsets in Sections 2.2 and 5.2, but the intuition above should be enough to understand the modelling of A&A, described in the next section.

2.2 Asynchronous Semantics

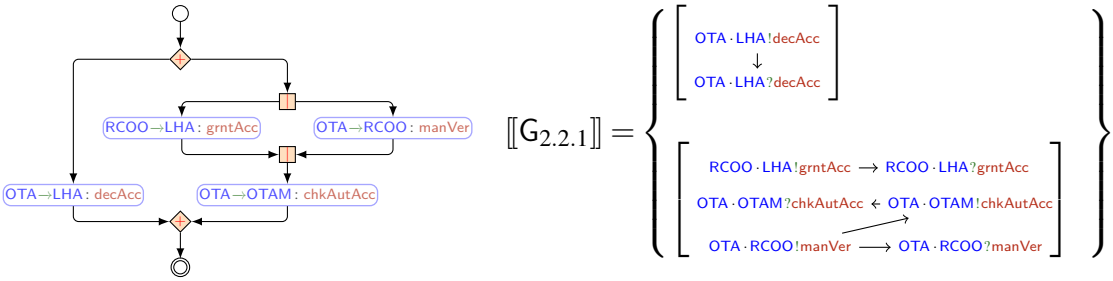
The asynchronous semantics of a g-choreography is modelled using *partially ordered multisets* (pomsets) [34]. A pomset consists of a finite set of events labelled by actions, together with a partial order describing causal dependencies between them. Concurrent events are not ordered, making pomsets a natural semantic model for asynchronous communication, where multiple interactions may occur independently. We used pomsets to reason about closure and realisability properties of choreographies. Essentially, a pomset r corresponds to a partial order $\leq_r$, called *happens-before* relation, that specifies the causal dependencies among some *events*; if $e \leq_r e'$ then e' causally depends e in the pomset r. Instead of the formal definition of pomsets of g-choreographies (which can be found in [6, 24]), we give an intuitive presentation of the main concepts.

Our pomsets order events labelled by communication actions. More precisely, actions of the form $\mathsf{A}\cdot\mathsf{B}!\mathsf{m}$ label events where A sends message m to B, while actions of the form $\mathsf{A}\cdot\mathsf{B}?\mathsf{m}$ label events where B receives message m from A. The semantics of a g-choreography G is then given by associating the set of pomsets $[\![\mathsf{G}]\!]$ corresponding to the resolutions of each choice in G. This semantics can be defined by induction on the syntactic structure of G. The base case being $\mathsf{A}\to\mathsf{B}:\mathsf{m}$ for which $[\![\mathsf{A}\to\mathsf{B}:\mathsf{m}]\!]$ is a singleton set containing a pomset where an event labelled with $\mathsf{A}\cdot\mathsf{B}!\mathsf{m}$ (dubbed *output event*) precedes the one labelled $\mathsf{A}\cdot\mathsf{B}?\mathsf{m}$ (dubbed *input event*). For the inductive cases, the semantics of a choice $\mathsf{G} + \mathsf{G}'$ is the union of the semantics of G and G' (that is $[\![\mathsf{G} + \mathsf{G}']\!] = [\![\mathsf{G}]\!] \cup [\![\mathsf{G}']\!]$); the semantics of $\mathsf{G};\mathsf{G}'$ merges the pomsets obtained by the Cartesian product of the pomsets of $[\![\mathsf{G}]\!]$ and $[\![\mathsf{G}']\!]$, adding dependencies that force events of the same participant in G to precede those in G'; the semantics of $\mathsf{G} \mid \mathsf{G}'$ is obtained by merging the pomsets obtained by the Cartesian product of $[\![\mathsf{G}]\!]$ and $[\![\mathsf{G}']\!]$, without adding any dependencies. Finally, the semantics of iteration is given by taking the union of the semantics of the finite

unfoldings of G, namely $[\![\mathit{repeat}\ \ \mathsf{G}]\!] = \bigcup_{n>0} \underbrace{[\![\mathsf{G}]\!]; \ldots; [\![\mathsf{G}]\!]}_{n-\text{times}}$.

Example 2.2.1. *Consider the choreography (inspired by A&A)* $\mathsf{G}_{2.2.1} = \mathsf{G} + \mathsf{G}'$, *where* $\mathsf{G} = \mathsf{OTA}\to\mathsf{LHA} : \mathsf{decAcc}$ *and* $\mathsf{G}' = (\mathsf{RCOO}\to\mathsf{LHA} : \mathsf{grntAcc} \mid \mathsf{OTA}\to\mathsf{RCOO} : \mathsf{manVer})$; $\mathsf{OTA}\to\mathsf{OTAM} : \mathsf{chkAutAcc}$.

The graphical representation of the g-choreography and its asynchronous semantics are as follows:



The g-choreography $\mathsf{G}_{2.2.1}$ *comprises two branches in a choice. The only interaction in the left branch is* $\mathsf{OTA}\to\mathsf{LHA} : \mathsf{decAcc}$, *while the right branch consists of the parallel interactions* $\mathsf{RCOO}\to\mathsf{LHA} : \mathsf{grntAcc}$ *and* $\mathsf{OTA}\to\mathsf{RCOO} : \mathsf{manVer}$, *followed by the interaction* $\mathsf{OTA}\to\mathsf{OTAM} : \mathsf{chkAutAcc}$.

Hence, $[\![\mathsf{G}_{2.2.1}]\!]$ *consists of a pomset where* LHA *must consume the* decAcc *from* OTA *and another where participants* LHA *and* RCOO *may receive messages* grntAcc *and* manVer *in any order. Note that the sending of message* chkAutAcc *depends only on the sending of message* manVer, *since they are the only interactions in different components of the sequential composition done by the same participant.* ◇

A *linearisation* of a pomset r is a sequence $\{e_i\}_i$ of its events that respects the happens-before relation of r, namely if $i < j$ then $e_j \not\leq_r e_i$. The language of a g-choreography is the union of the sequences of labels of all the *linearisations* of the pomsets of its semantics.

A g-choreography among participants $\mathsf{A}_1, \ldots, \mathsf{A}_n$ is implemented via a *communicating system* [13], that is an assignment to each participant A_i of a *communicating finite-state machine* (CFSM) [13] (i.e., a finite-state automaton whose transition labels are communication actions of A_i). In our asynchronous model of communicating systems, a configuration is a tuple $(q_1, \ldots, q_n, M)$, where q_i is the current state of the CFSM of A_i and M is an unordered multiset of (pending) messages. The firing of a transition $q_i \xrightarrow{\mathsf{A}_i \cdot \mathsf{B}\,!\,\mathsf{m}} q_i'$ yields the new configuration $(q_1, \ldots, q_i', \ldots, q_n, M \uplus \{\mathsf{A}_i \cdot \mathsf{B}\,!\,\mathsf{m}\})$, where $\uplus$ denotes multiset union. If $M = M' \uplus \{\mathsf{B} \cdot \mathsf{A}_i\,!\,\mathsf{m}\}$ then the firing of a transition $q_i \xrightarrow{\mathsf{B} \cdot \mathsf{A}_i\,?\,\mathsf{m}} q_i'$ yields the new configuration $(q_1, \ldots, q_i', \ldots, q_n, M')$. The language of a system of CFSMs is the set of all sequences of labels on runs starting from its initial state.

Some g-choreographies are not properly implementable. For example, in $\mathsf{G}_{2.2.1}$ (from Example 2.2.1), participants must agree on whether to take the left or the right branch. However, RCOO is not involved in the left branch, and can decide autonomously to move on the right branch sending message grntAcc, while participant OTA may send decAcc taking the left branch. This leads to an erroneous computation, not expected from the choreography $\mathsf{G}_{2.2.1}$.

Formally, a g-choreography G is *realisable* if there exists a communicating system S such that $\mathcal{L}(S) = \mathcal{L}(\mathsf{G})$, where $\mathcal{L}(\mathsf{G})$ is the set of all linearisations of the pomsets in $[\![\mathsf{G}]\!]$. This ensures that every run of S conforms to at least one linearisation of a pomset of the g-choreography and vice-versa.

Two sufficient closure conditions for the realisability of a set of pomsets under the asynchronous semantics have been described in [24]. We introduce some auxiliary notation to state them (see Example 2.2.2 for concrete uses of the notations below). A pomset r_1 is a prefix of a pomset r_2 if its events are a subset of the events of r_2 and it is downward closed w.r.t. the happens-before relation of r_2 (i.e., if $e \leq_{r_2} e'$ and $e' \in r_1$ then $e \in r_1$) and equipped with the projection of the happens-before relation of r_1. Under the asynchronous

semantics, a pomset is *well-formed* if [35]:

- each output event $\mathsf{A} \cdot \mathsf{B}!\mathsf{m}$ (resp. input event $\mathsf{A} \cdot \mathsf{B}?\mathsf{m}$) has at most (resp. exactly) one immediate successor input event $\mathsf{A} \cdot \mathsf{B}?\mathsf{m}$ (resp. immediate predecessor output event $\mathsf{A} \cdot \mathsf{B}!\mathsf{m}$);
- whenever an event e is an immediate predecessor of another event e', then e and e' are labelled either by communication actions of the same participant or by matching output and input actions; and
- for any two distinct outputs e and e' with the same label and e' causally depending on e, no immediate-successor matching input of e' may precede any immediate-successor matching input of e.

The projection of a pomset r onto a participant A, denoted $r\!\upharpoonright_{\mathsf{A}}$, is the subpomset consisting of all events in r that are labeled with communication actions involving A. Given a set of pomsets R over participants $\mathcal{P}$, we define the set of *branch assignments on* R as the set Π_R of functions such that $\pi(\mathsf{A}) \in \{r\!\upharpoonright_{\mathsf{A}} \mid r \in R\}$ for $\mathsf{A} \in \mathcal{P}$. Intuitively, a branch assignment $\pi \in \Pi_R$ chooses, for each to participant $\mathsf{A} \in \mathcal{P}$, a pomset $r \in R$ (corresponding to a resolution of a g-choreography) and assigns $r\!\upharpoonright_{\mathsf{A}}$ to A. Note that π can choose different resolutions for different participants. We use $\cup_{\mathsf{A}}\pi(\mathsf{A})$ for the pomset obtained by merging the pomsets in the codomain of π. Finally, the *inter-participant closure* of a pomset, denoted $\square(r)$, is the set of all pomsets that can be constructed by adding dependencies to connect individual output events with their corresponding input events across participants. The resulting pomsets in $\square(r)$ represent all valid ways of linking outputs to inputs. A pomset r is *more permissive than* another pomset r' if r has the same events and labels of r' and its happens-before relation is a subset of the one of r' (namely, r imposes fewer causal dependencies than r'). The closure conditions CC2 and CC3 below jointly guarantee relisability. Intuitively, CC2 requires that if a set of executions (i.e., a pomset) cannot be taken apart by any participant from the executions (i.e., pomsets) of

R, then those executions must be part of R.Intuitively, CC3 requires that if a set of executions (i.e., a pomset) can be consistently represented as a prefix R' of another pomset in R, indicating that some participants will terminate earlier while matching the executions of R, then those prefixes R' must be part of R.

Definition 7 (Closure conditions)**.** *A set of pomsets R* satisfies CC2 *if for all well-formed $r \in \{\Box(\cup_{\mathsf{A}} \pi(\mathsf{A})) \mid \pi \in \Pi_R\}$ there is $r' \in R$ that is at least as permissive as r; R* satisfies CC3 *if for all R' set of prefixes of R and all well-formed $r \in \{\Box(\cup_{\mathsf{A}} \pi(\mathsf{A})) \mid \pi \in \Pi_{R'}\}$, there is a prefix r' of a pomset in R such that r' is at least as permissive as r.*

The closure conditions in Definition 7 are adapted from the realisability criteria for MSCs [35] and avoid the explicit computation of the languages of the choreography.

Example 2.2.2. *We illustrate our closure conditions on the semantics of $\mathsf{G}_{2.2.1}$ from Example 2.2.1. Since $[\![\mathsf{G}_{2.2.1}]\!]$ consists of two pomsets involving participants* RCOO*,* LHA*,* OTA*, and* OTAM*}, Π_R contains 2^4 functions, one for each possible combination of the projections of these pomsets on the four participants.*

It is straightforward to see that only two functions of Π_R have a well-formed inter-participant closure: the ones where all participants have chosen the same pomset of R. For example, in one of the other cases, if the participant RCOO *chooses the first pomset and* OTA *chooses the second one, the input* $\mathsf{OTA} \cdot \mathsf{RCOO}?\mathsf{manVer}$ *has no corresponding output. Since the two well-formed pomsets coincide with the choreography semantics, the CC2 condition is met.*

However, CC3 is not satisfied. In fact, let $\pi(\mathsf{RCOO}) = [\mathsf{RCOO} \cdot \mathsf{LHA}!\mathsf{grntAcc}]$ *(i.e., prefix of the first branch),* $\pi(\mathsf{LHA}) = [\mathsf{RCOO} \cdot \mathsf{LHA}?\mathsf{grntAcc}]$ *(i.e., first branch),* $\pi(\mathsf{OTA}) = [\mathsf{OTA} \cdot \mathsf{LHA}!\mathsf{decAcc}]$ *and* $\pi(\mathsf{OTAM}) = []$ *(i.e., both second branch). The inter-participant closure yields the well-formed pomset:*
$r = [\mathsf{RCOO} \cdot \mathsf{LHA}!\mathsf{grntAcc} \rightarrow \mathsf{RCOO} \cdot \mathsf{LHA}?\mathsf{grntAcc} \quad \mathsf{OTA} \cdot \mathsf{LHA}!\mathsf{decAcc}]$*; However no prefix of pomsets in R is more permissive than r. Pomset r (as any other where*

RCOO *chooses one branch and* OTA *chooses the other one) demonstrate the lack of direct or indirect (via other participants) communication between* OTA *and* RCOO *to select the proper branch.* ◇

Note that condition CC3 does not imply condition CC2, as proved by the counterexample in [36].

Chapter 3

Case Study

3.1 A&A: Natural Language Description

Design and development of distributed communication systems is very complex. Researchers and practitioners consider the choreography approach suitable to verify the communication properties of distributed systems to avoid issues such as deadlocks at the design level of the software development [37–40]. The key purpose of choreographies is to enact the principle of correctness-by-construction [7], which ensures that choreographies satisfying suitable conditions are free from communication errors such as deadlocks. We apply the choreographic approach to a A&A on healthcare management. Health is a fundamental right of the individual. The Emilia-Romagna region provides regional health care services to protect the fundamental rights of their citizens. It also ensures the quality, safety, and transparency of health services through the process for authorisation and accreditation of public and private health structures [41], turned on in 1998 and nowadays regulated by the Regional Law of 6 November 2019, n. 22 [42] .

The healthcare authorisation procedure aims at ensuring the quality of healthcare services and assistance to the citizens. The authorisation guarantees the structural, organisational, technological, and safety requirements of the patients

in the public and private healthcare facilities in the Emilia-Romagna region in Italy. The authorisation to operate the healthcare infrastructure is issued by the Mayor of the municipality. In contrast, the accreditation gives organisations, healthcare structures, and professionals, already in possession of healthcare authorisation, the status of "subject suitable for providing services on behalf of the National Health Service". It provides for the possession of additional requisites that refer to the quality of health care and the related evaluation methods. It is granted by the *Regional Coordinator of Authorisation and Accreditation* with subsequent verification of the *Technical Accreditation Body* aimed at ascertaining the possession or maintenance of the requirements. We have identified three protocols in this process, namely regional coordination for authorisation and accreditation, authorisation of health services, and accreditation of health services. In this A&A, we will focus on the regional coordination for authorisation and accreditation which assists in providing consistency and homogeneity in the exercise of the functionalities of authorisation and accreditation processes. Initially, we began formalising the A&A based on the description in the law, which is in natural language. Since some aspects of the law are implicit and not fully detailed, the initial protocol design, reported in this section, does not perfectly match the actual protocol. However, first analysis were done on such a model, hence we report them. Results would not be too different in the correct model. Subsequently, we consulted one of the actual participants in the protocol (with the role of Technical Accreditation Body Manager), and this allowed us to improve the formalisation. We will present the revised protocol in Section 5.1. Ten participants are involved in this protocol, described below. In order to simplify the match between our models, which use English names for participants, and the original document, which is in Italian, we report the translations in Table 3.1. The participants are:

- General Directorate of Health (GDH): The highest authority in the process of authorisation and accreditation, it instructs the other sub-ordinate participants to perform their tasks to ensure consistency within the autho-

risation and accreditation, to maintain the organisation rationalisation, and to maintain the homogeneity to perform the functions.

- Regional Coordinator (RCOO): It is the main authority to conduct the different functions to issue authorisation and accreditation to healthcare structures.
- Authorisation Commission (AC): It establishes itself in the public health departments of the local health unit to ensure the homogeneity in the assessment of authorisation and it also identifies the priority of criteria to carry out checks for the authorisation.
- Local Health Authority (LHA): It is the healthcare organisation that is seeking authorisation and accreditation.
- Registry of Healthcare Structures (RHS): It stores the data for the accreditation and it provides the data upon request from RCOO.
- Technically Accrediting Body (OTA): It performs the accreditation checks to ensure impartiality, transparency, and autonomy in the activities of the management.
- OTA Manager (OTAM): It is a manager expert in assessing the quality of the healthcare management system.
- Evaluators (EV): They are qualified and trained professionals from the public and private national health services that are situated within the OTA to pursue the technical assessment of the accreditation.
- Expert Technicians (ET): They perform further technical investigations on the LHA.
- Regional Council (RC): It identifies the RCOO among the managers of its own services to ensure homogeneity and consistency in the process. It also decides how to manage the EV on the proposal of the OTAM.

Table 3.1: Participant Name in Italian and Corresponding English Translation

Participants Name (Italian)	**Participants Name (English)**
Direzione Generale Competente in Materia di Sanità	General Directorate of Health (GDH)
Giunta Regionale	Regional Council (RC)
Coordinatore Regionale per l'Autorizzazione e l'Accreditamento	Regional Coordinator (RCOO)
Commissioni per l'Autorizzazione	Authorisation Commission (AC)
Aziende Usl	Local Health Authority (LHA)
Anagrafe Regionale delle Strutture Sanitarie	Registry of Healthcare Structures (RHS)
Organismo Tecnicamente Accreditante	Technically Accrediting Body (OTA)
Responsabile dell'OTA	OTA Manager (OTAM)
Elenco dei Valutatori	Evaluators (EV)
Tecnici Esperti	Expert Technicians (ET)

In a distributed communicating system, all the participants interact through message passing. We now give an informal description of A&A, to be formalised in the next sections. The protocol includes an initialisation phase, where roles and resources are assigned, and then 4 concurrent loops corresponding to different functionalities of the RCOO. The main loop is the actual process to check accreditation of LHA. In A&A, the RCOO will grant accreditation to the public or private LHA based on the verification of the OTA. The GDH notifies the RC to identify the RCOO among the managers of its own services. Subsequently, the GDH provides resources to the RCOO to perform their functions and it also instructs the RCOO to pursue verification of specific functions to evaluate the LHA for further continuation of the accreditation. The RCOO formulates proposals on which data the RHS needs to collect, and on how it should work, to the RC. Then, the RC sends the criteria established for the topics above to the RHS. The RCOO coordinates with the AC to check whether existing authorised LHA maintain prerequisites and satisfy additional criteria to maintain the quality of healthcare. It also prepares plans for issuing, renewing, and monitoring accreditation and proposes them to the GDH. The RCOO maintains relations with the services of the GDH in order to guarantee the connection between its

own functions and the policies and skills of the area. Moreover, it mandates the OTA to conduct verification to check whether existing authorised LHA are maintaining the prerequisites to provide safe health facilities. Besides, the OTA delegates the actual checks to an OTAM. The OTAM prepares the verification report of the LHA and it also conducts further technical investigation with support from the experts, such as EV and ET. Thereafter, the OTAM sends the report to the RCOO. The RCOO prepares the proposal for renewal, refusal, or issuing of the accreditation based on that report and submits it to the GDH to take the necessary action on the LHA.

3.2 A&A: BPMN Collaboration Diagram

Since the specification of A&A, briefly described in the previous section, is given in natural language and hence necessarily informal, transforming it directly into a choreographic model is complex and error-prone. Therefore, we decided to model the protocol first using a BPMN collaboration diagram [16], which is closer to the informal specification since they both focus on participants, activities and processes, while choreographies focus on message passing. However, moving to a BPMN collaboration diagram allows one to move from an informal to a semi-formal specification, and to explicitly introduce choices and message passing communications, key to then move to choreographies. The fact that the representation of the BPMN collaboration diagram is graphical and closer to the specification also allows an easier interaction with domain experts to ensure that the model actually captures the intended protocol. As a second step, we then converted the BPMN collaboration diagram into a CA [14] to enable the verification of the correctness of its structure and behavior using the Corinne tool [17, 18]. We transformed the natural language description of the healthcare system into a BPMN collaboration diagram manually. The BPMN collaboration diagram was initially constructed based on the available documen-

tation and then discussed with a stakeholder involved in the accreditation process (OTAM). The stakeholder confirmed that the model was understandable and largely accurate, suggesting only minor revisions to better reflect operational details. These changes did not modify the communication patterns captured by the model and therefore did not affect the subsequent choreography-based modelling and verification.

A BPMN collaboration diagram is a semi-formal specification with a wide variety of expressive and flexible features to ease the representation of complex systems. In this section, we present a fragment of A&A, available in Figure 3.1, to give to the reader an intuition about the transformation of A&A from natural language to BPMN specification. The whole A&A model is available in [43]. The BPMN collaboration diagram consists of different participants [44] that represent a specific role or entity that is involved in a business process. Each participant has its own internal processes, and participants interact through message passing. Our example in Figure 3.1 involves six participants, namely RC, RCOO, OTA, OTAM, EV, and ET. The process diagram describes the details of a business process involving them. Notably, it provides a visual representation that gives a clear understanding of the flow of information and the sequence of activities of the business process. In Figure 3.1, for compactness, we detail the process diagram of the participant OTAM, while leaving not specified the behavior of other participants. Below, we will describe the different BPMN features and symbols when they occur in our description of the A&A, while referring to [16] for a more comprehensive description.

The participants communicate among themselves through message passing. Therefore, the *message event* (double circle containing an envelope) indicates the sending or receiving of a message to/from another process or participant. A dashed line connects the send and the receive. The OTAM is first involved in the protocol when the OTA appoints it for carrying out the authorisation and accreditation checks. Therefore, the OTA notifies the OTAM to conduct checks

on existing authorisations through the message event on the left.

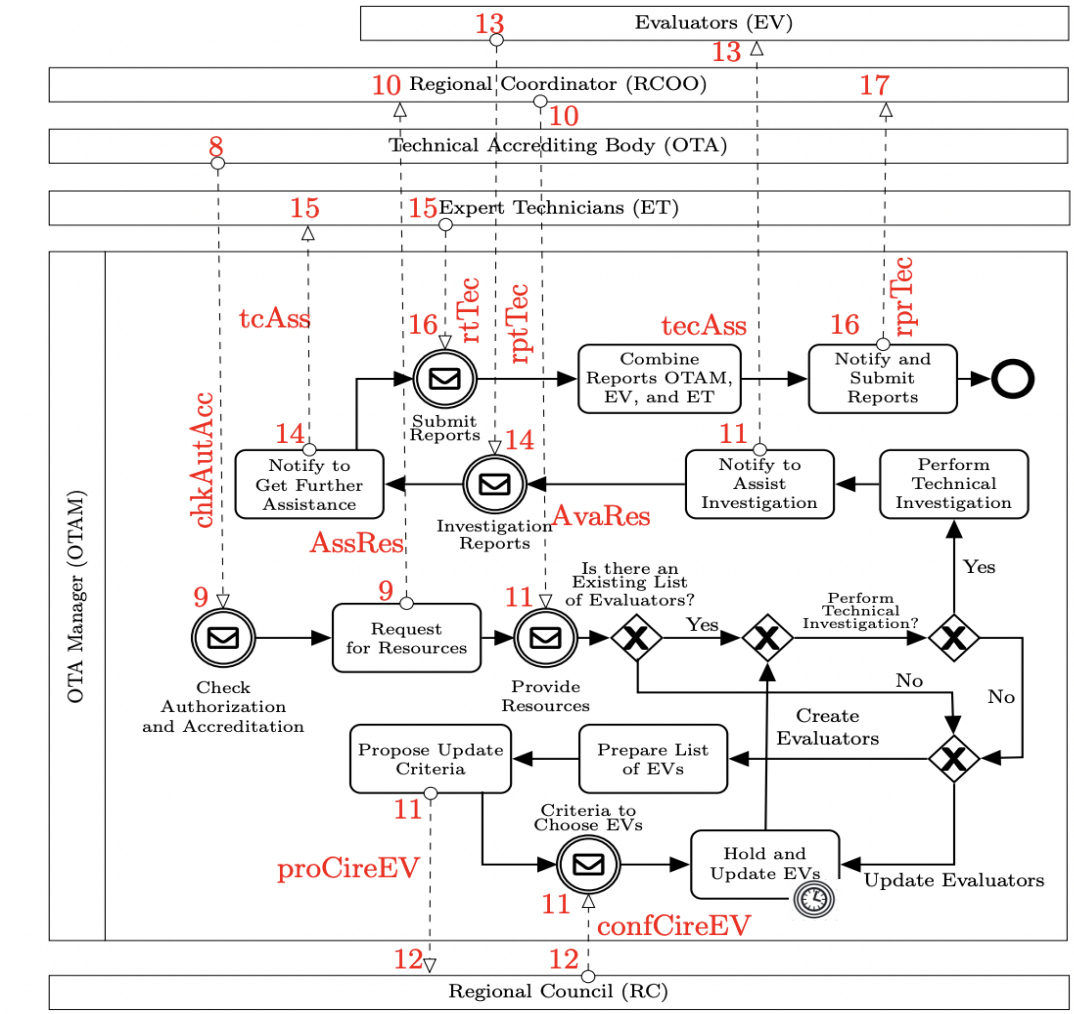


Figure 3.1: BPMN Collaboration Diagram: A Fragment of A&A.

A process is a collection of multiple tasks: in BPMN a single activity within a process is called a *task*. The OTAM, after receiving notification from the OTA, performs the task **Request for Resources**, where the OTAM requires the RCOO to provide resources to perform its functions. Note that the dashed arrow from the activity denotes that as part of the activity a message is sent. Thereafter, the RCOO provides the required resources to the OTAM via a **Provide Resources** message. In BPMN, an *Exclusive Gateway or XOR Gateway* (diamond containing an X) is a decision gateway used to model a point in the process

where the flow can take one of several mutually exclusive paths. The decision about which path to take is based on the evaluation of conditions associated with each outgoing sequence flow. The OTAM, after getting resources from the RCOO, checks whether the list of EVs exists. If not, then it first creates the list of EVs. Otherwise, the OTAM needs to **Perform Technical Investigation** for accreditation or needs to **Update Evaluators List** for assistance with the technical investigation. Here, we use an exclusive gateway to represent the choice of the path. The detailed explanations of the two paths are as follows:

- **Perform Technical Investigation:** The OTAM carries out the technical investigations, and expresses the technical judgment of competence by verifying the possession and maintenance of the accreditation requirements.
- **Update Evaluators List:** The OTAM holds and updates the list of EVs.

For the technical investigation for the checks of authorisation and accreditation, the OTAM may need assistance from experts such as EV and ET sequentially, as follows:

1. The OTAM performs the checks by making use of the EVs. The OTAM **Notify to Assist Investigation** to the EVs. The EVs perform the technical investigation and submit to the OTAM the technical **Investigation Report**.
2. The OTAM, after receiving the investigation report from the EVs, needs to perform further investigation with the assistance of the ETs. Therefore, it **Notify to Get Further Assistance** to the ETs. The ETs assist and **Submit Report** to the OTAM.

The OTAM **Combine Reports of** OTAM**,** EVs**, and** ETs after performing all necessary checks for authorisation and accreditation. Thereafter, (s)he **Notify and**

Submit Report to the RCOO and terminates the process. To **Create and Update Evaluators List**, the OTAM performs the tasks sequentially i.e., **Prepare List of Evaluators**, **Propose Update Criteria**, **Update Evaluator List**. Moreover, the list of EVs gets updated periodically.

Chapter 4

Analysis with Corinne

4.1 A&A: Choreographic Automaton

We use Choreographic Automata (CAs) [14] to model the message-passing behavior of A&A. This will allow us to verify the correctness of such behavior. The key concept of choreographic approaches is to describe the interactions among participants from a global viewpoint. In CAs, finite state automata are used, and a transition of the automaton models a message passing communication. More precisely, the interaction among two participants is represented as a label A→B : m, where participant A sends message m to participant B, which in turn receives it [14]. In this section, we will discuss the transformation of A&A from BPMN collaboration diagram to CA. We transformed the BPMN collaboration diagram into a CA manually. Note that we have a correspondence between dashed arrows representing message flow in the BPMN Collaboration Diagram and interactions in the corresponding CA.

We present the CA of A&A in Figure 4.1. To ease the comparison with the BPMN collaboration model, identifiers of states from the CA as well as labels of the transitions are also reported in the BPMN collaboration diagram, in red. Remember however that only part of the BPMN model is in Figure 3.1, while the whole model is in [43]. Let us start the description from the initial state

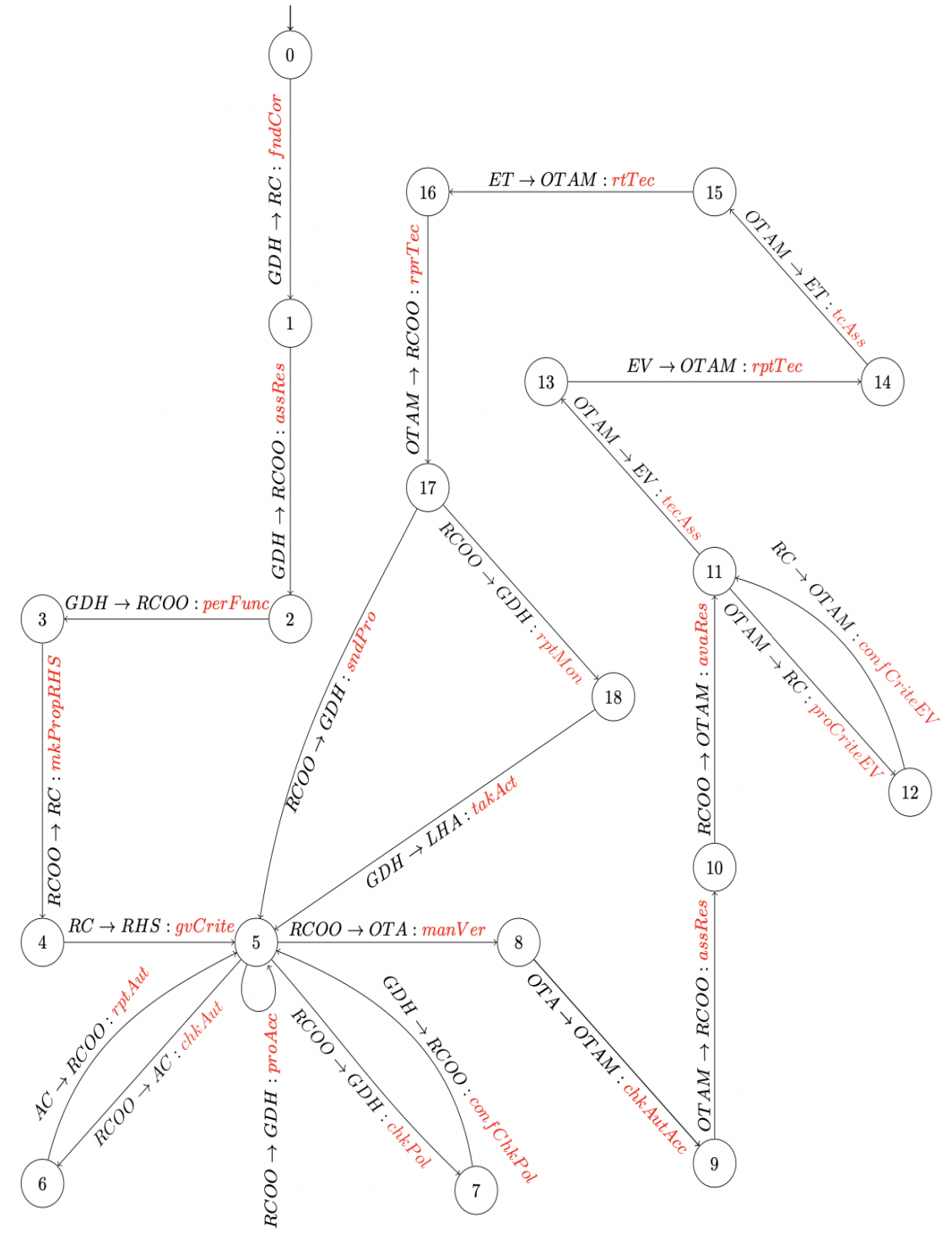


Figure 4.1: Choreographic Automaton for Regional Coordination for Authorisation and Accreditation.

(state 0 in the top-left). GDH proposes RC to identify a RCOO with transition GDH→RC : fndCor, to maintain consistency, organisational rationalisation, and homogeneity of authorisation and accreditation. Further, the GDH notifies the RCOO that resources have been assigned GDH→RCOO : assRes to execute the authorisation and accreditation procedure. Moreover, the GDH requires the RCOO to perform the required functions GDH→RCOO : perFunc for accreditation. The RCOO requires adequate information to perform the tasks for accreditation and authorisation, which will be managed by the RHS. Hence, the RCOO proposes to the RC with message RCOO→RC : mkPropRHS which data the RHS should manage and how it should work. As a result, the RC gives the criteria to the RHS with message RC→RHS : gvCrite on which data to manage and how. Afterward, the RCOO may take 4 alternative actions:

1. (loop involving state 6) to ask the AC to check an existing authorisation RCOO→AC : chkAut of some LHA for further continuation. The AC will respond with the authorisation report AC→RCOO : rptAut message to the RCOO;

2. (loop involving state 7) to request the GDH for consultation to prepare a monitoring policy RCOO→GDH : chkPol, which will trigger a reply from the GDH confirming the consultation, GDH→RCOO : confChkPol;

3. (self-loop at state 5) to propose to the GDH the regional planning for issuing, renewing and monitoring accreditation RCOO→GDH : proAcc;

4. (transition to state 8) to ask the OTA to start the actual procedure for accreditation, RCOO→OTA : manVer.

In this last case, the OTA notifies the OTAM to check authorisation and accreditation OTA→OTAM : chkAutAcc for checking the quality of the healthcare management systems of some LHA (this starts the part represented in BPMN in Figure 3.1). The OTAM requests the RCOO to assign the corresponding

resources OTAM→RCOO : assRes, and the RCOO makes such resources available RCOO→OTAM : avaRes to the OTAM. Afterward, the OTAM may take 2 alternative paths: (1) (loop involving state 12) To manage the list of EVs: while this procedure is quite complex it is mostly local to the OTAM, hence we only have a loop composed by a pair of interactions, where the OTAM sends to the RC the proposal OTAM→RC : proCriteEV message about the criteria to manage the EVs, and the RC answers to the OTAM with message RC→OTAM : confCriteEV to confirm the proposed evaluation criteria; or else (2) the OTAM asks the EVs for technical assistance with message OTAM→EV : tecAss and the EVs reply with a technical report EV→OTAM : rptTec.

In the last case, after receiving the technical report from the EVs, the OTAM performs a similar message exchange with the ETs (messages OTAM→ET : tcAss and ET→OTAM : rtTec). Note that from the informal specification it is not clear whether the two message exchanges should be sequential as in our model or concurrent. We will come back to this topic in the discussion. Further, the OTAM sends to the RCOO the results of the investigation with message OTAM→RCOO : rprTec. The RCOO may react in 2 ways (from left to right): (a) It sends to the GDH with message RCOO→GDH : sndPro a proposal for grant, renew, refusal, suspension, or revocation of the accreditation, based on the report received from the OTA, and it ends the protocol, or (b) It sends to the GDH with message RCOO→GDH : rptMon the report on the monitoring of the LHA, based on the OTA report. Upon further analysis, and based on the seriousness of the violation of the requisites of the LHA, the GDH notifies the LHA with message GDH→LHA : takAct the outcome of the evaluation. This may involve to revoke or suspend the accreditation, or to take necessary actions to fix some violations.

4.2 Validation

As a result of the modelling exercise above, we represented A&A in the form of CA. CAs are equipped with a theory [14] ensuring that systems obtained from their projection are well-behaved, provided that the CA is well-formed. Well-formedness can be automatically checked using the tool Corinne [17]. Below we introduce *well-formedness*, which is composed of *well-sequencedness* and *well-branchedness*, and we discuss its application to A&A. The *well-formedness* properties described in the Section 2.1.1. To simplify the analysis, we slightly extended Corinne. Indeed, Corinne has been conceived to iteratively improve a CA till it becomes well-formed, hence, in case the CA contains many violations of the conditions, Corinne only showed the first one. We are more interested in getting full feedback on a protocol to then discuss with domain experts about the issues. Thus, we extended the tool so that all the violations of the well-formedness conditions were shown in a single execution. We also did other minor improvements, and the updated version of Corinne is available on the Corinne repository [18].

By using the Corinne tool, we first checked well-sequencedness. A number of pairs of transitions were flagged as violating the condition, but interestingly they can be categorised in two classes, both involving state 5. On the one hand, transition from 4 to 5 with label RC→RHS : gvCrite (let us call it t_g) has no participant in common with any of the transitions exiting from state 5. On the other hand, transition from 18 to 5 with label GDH→LHA : takAct (let us call it t_a) has no participant in common with transitions from state 5 to either 8 or 6. The intuition behind the issue is similar in both the cases. State 5 is a state where participant RCOO decides which actions to take, but the two incoming transitions t_g and t_a do not involve him, hence the issue. There are two ways to fix the diagram, if we deem that it is important that the RCOO is notified about the end of t_g and/or t_a, then notification transitions should be

added. E.g., in the case of t_g, RHS should send a message to RCOO notifying him that criteria have been received. Vice versa, one can decide that transitions t_g and t_a should be concurrent to the following actions. In CA, this is done by explicitly duplicating transitions t_g and t_a so that copies of them could occur at any point in the following. This will represent them as concurrent actions, but would make the diagram unreadable. We will come back to this issue in Section 4.3.

We then tested well-branchedness. Here a much larger set of issues is raised, always related to condition 3. We describe one of the issues raised in detail, and then briefly comment on the others which are similar. Let us consider as an example the cycles from state 11: the one via 13, 14, 15, 16, 17, 18, 5, 8, 9, 10 involves multiple participants, while the one via state 12 only involves OTAM and RC. This is flagged as an error of well-branchedness, since other participants, say EV, are left waiting if the choreography always takes the cycle via 12. This is not a real issue here, since in practice the small cycle is taken a small number of times, and more importantly since the other participants are not actively waiting that the choreography involves them, but just react when invoked. Taking into account that participants may only be involved in some branches of the choreography takes the name of *late join* (or selective participation), and it has been studied in the literature [20], but it is not yet supported by Corinne. We will come back to this issue in Section 4.3.

Actually, all the other issues raised by Corinne are also related to late join, since the same issue happens in many other cases, e.g., between different cycles originating from state 5, or between the two paths from state 17 to 5. Notably, the main reason behind well-branchedness is to ensure that when there is a choice, participants which need to act differently in different branches are notified of the outcome of a choice. This always happens in A&A. Consider for instance participant GDH. It is involved in different branches of the choice from 5, and the sub-choice from 11. Indeed, at 7 it receives a message chkPol

from RCOO and then needs to react with message confChkPol, at 17 it receives a message sndPro but no reaction is expected from him, and at state 18 it receives a message rptMon and it is expected to react by sending message takAct to LHA. The GDH is not directly participating to the choice of which path to take from 5 and from 17, but it is always informed of the outcome by RCOO, which is in charge of deciding the path to be taken, and sends different messages to GDH in all the paths, hence GDH always knows how to react.

4.3 Lessons Learned

We discuss below a few lessons we learned from the modelling exercise and analysis above. They will probably apply to other case studies in the same area, and possibly also in other areas, but this will need to be confirmed by further studies.

Gap between laws and choreographies: Laws describe protocols in terms of roles, responsibilities of each role, and activities performed by each role. There are some mentions of message passing, but it is frequently implicit and not detailed. Explicit mentions are, e.g., "The global results are sent to RCOO to proceed with the relevant obligations" (translated from [42, Art. 13, paragraph 3.b], resulting in the transition from state 13 to 14). Implicit mentions are, e.g., "The organisational methods, human and instrumental resources to be assigned to the RCOO, for performing the functions referred to in paragraph 3, are defined by the GDH." (translated from [42, Art. 3, paragraph 2]), which we modelled as the transition from state 1 to state 2. In order to better match the activities described by the laws, and the message passing modelled by the choreography, it makes sense to use an intermediate model such as a BPMN collaboration diagram, which describes both the activities and the message exchanges.

This will be even more needed if multiparty session types [1, 7] were used instead of CAs, since they impose more constraints on the model than CAs.

Importance of late join: the only issues raised by Corinne related to well-branchedness are due to the missed support for late join in Corinne. While historically approaches in the area of choreographies and session types assume that all the participants are aware of when the choreography starts and how it progresses, in reality, as already mentioned in [20], most of the participants simply react by taking action when involved by other participants, and do not care if they are not involved in some execution. Hence, we believe that late join [20] is fundamental to reason on real life protocols, and would deserve more attention by the community. In our case, supporting late join in Corinne is a main item for future work. We mention that there are also approaches [45, 46] that allow for late join, but they require special actions for joining a protocol. In A&A, it seems joining as a result of receiving a normal message more closely matches reality. Moreover, the approach based on g-choreographies avoids the issue of late join since most of the problems that described to *late join* disappear when using the PomCho+ tool. Nonetheless, there is still an issue between the two paths from state 17 to state 5. This issue is not a real issue in the protocol but an error while modelling. In this regard, PomCho+ highlights its value in detecting subtle modelling flaws that might go undetected under less exact semantics by exposing inconsistencies already present in the model rather than reintroducing *late join* problems. Thus we will turn to this approach in Chapter 5.

Parallel protocols: in practice, many actors carry on multiple protocols at the same time. This is the case of RCOO in A&A, where each activity corresponds to one of the loops from state 5. We modelled these activities using a choice, but in practice these activities can happen together, hence our model does not fully capture the actual behavior. However,

such a model would require in CAs to explicitly represent all the possible interleavings, which would make the model very difficult to manage and reason about, due to the combinatorial explosion of states. In order to avoid such problem one could allow for an explicit representation of parallel composition, as for instance in g-choreographies [6] and BPMN collaboration or choreography diagrams. Indeed, in Chapter 5 we use g-choreographies to solve this problem.

Creation of new participants: somehow surprisingly, creation of new participants occurs quite often in A&A, in the sense that persons are given some role that they then use in the protocol. For instance, the RC nominates the RCOO, and then GDH interacts with the newly appointed RCOO. Currently, explicit creation of new roles is not supported by CAs (and the same holds for most formalisms in the area), hence errors such as a participant having to interact before its creation are not highlighted.

4.4 Publication

Choreographic Automata: A Case Study in Healthcare Management*

Sourabh Pal[1][0000−0001−6604−387X], Ivan Lanese[1][0000−0003−2527−9995], and Massimo Clo[2]

[1] Olas Team, University of Bologna/INRIA, Bologna, Italy
{sourabh.pal2,ivan.lanese}@unibo.it
[2] Area ICT e Transizione digitale dei servizial cittadino - Direzione Generale Cura della Persona, Salute e Welfare - Regione Emilia Romagna, Italy
massimo.clo@regione.emilia-romagna.it

Abstract. Choreographic models in general, and Choreographic Automata (CA) in particular, can be used to analyze and validate communicating systems. We applied CAs to a case study in healthcare management, the procedure for accreditation and authorization of public and private healthcare structures in the Emilia Romagna region (Italy). We formalized the procedure first using a BPMN collaboration diagram as intermediate step, and then using CAs. The tool Corinne showed a few issues in the formalized model, but it turned out that such issues were due to too strict requirements posed by the theory underlying Corinne. This gave us useful feedback for future improvements of Corinne and its underlying theory.

Chapter 5

Analysis with PomCho

5.1 A&A: Process Management in Italian Healthcare

The analysis of the A&A in Chapter 4 uses CAs and the Corinne tool. A main drawback of CAs (and consequently of Corinne) is that concurrency is modelled by explicitly representing all interleavings. This makes both the diagrammatic representation and the analysis quickly become unfeasible for highly concurrent scenarios, such as the one shown in Figure 4.1. In contrast, *g-choreographies* support parallel composition in both its diagrammatic representation and underlying theory. In this chapter, we present and analyse the CA in Figure 4.1 in the form of a *g-choreography*, as it naturally supports parallel composition. The protocol describes the interactions among eleven participants, namely the actors and organisations involved in the process. The analysis of A&A based on CAs [14] and the Corinne tool [17, 18] in [47] cannot model concurrent flows of execution in a suitable way (described in Section 4.3). Here, we therefore use *global-choreographies* (g-choreographies) to model A&A and the PomCho tool for the analysis since they natively support concurrency.

The g-choreography modelling A&A takes the form $\mathsf{G}_I;(\mathsf{G}_T \mid \mathsf{G}_A)$ where (As-

sume that $\_;\_$ takes precedence over $\_ + \_$.)

$$
\begin{aligned}
\mathsf{G}_I = \;& \mathsf{GDH}\to\mathsf{RC} : \mathsf{fndCor}; \mathsf{GDH}\to\mathsf{RCOO} : \mathsf{assRes}; \mathsf{GDH}\to\mathsf{RCOO} : \mathsf{perFunc}; \\
& \mathsf{RCOO}\to\mathsf{RC} : \mathsf{mkPropRHS}; \mathsf{RC}\to\mathsf{RHS} : \mathsf{gvCrite}
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{G}_T = \;& \mathit{repeat}\ \ \mathsf{RCOO}\to\mathsf{GDH} : \mathsf{proAcc} \\
& \mid \mathit{repeat}\ \ (\mathsf{RCOO}\to\mathsf{AC} : \mathsf{chkOut}; \mathsf{AC}\to\mathsf{RCOO} : \mathsf{rptinvs}) \\
& \mid \mathit{repeat}\ \ (\mathsf{RCOO}\to\mathsf{GHD} : \mathsf{chkPol}; \mathsf{GHD}\to\mathsf{RCOO} : \mathsf{confChkPol}) \\
& \mid \mathit{repeat}\ \ (\mathsf{RC}\to\mathsf{OTM} : \mathsf{proCriteEv}; \mathsf{RC}\to\mathsf{OTAM} : \mathsf{confCriteEv})
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{G}_A = \;& \mathit{repeat}\big(\mathsf{LHA}\to\mathsf{GDH} : \mathsf{seekAcc}; \mathsf{GDH}\to\mathsf{RCOO} : \mathsf{forAcc}; \\
& \qquad \mathsf{RCOO}\to\mathsf{HAS} : \mathsf{perInvs}; \mathsf{HAS}\to\mathsf{RCOO} : \mathsf{rtpinvs}; \big(\\
& \qquad\quad \mathsf{RCOO}\to\mathsf{LHA} : \mathsf{decAcc} \\
& \qquad\quad + \\
& \qquad\quad \big(\mathsf{RCOO}\to\mathsf{LHA} : \mathsf{gnrtAcc} \mid \mathsf{RCOO}\to\mathsf{OTA} : \mathsf{manVer}\big); \\
& \qquad\quad \mathsf{G}'_A; \\
& \qquad\quad \big(\mathsf{RCOO}\to\mathsf{GDH} : \mathsf{sndPro} \\
& \qquad\quad\quad + \\
& \qquad\quad\quad \mathsf{RCOO}\to\mathsf{GDH} : \mathsf{rptMon}; \mathsf{GDH}\to\mathsf{LHA} : \mathsf{takAct}\big)\big)\big)
\end{aligned}
$$

Figure 5.1 reports the g-choreography of A&A using the visual representation of g-choregraphies (cf. Section 2.1.2). Both the syntactic and visual representation of A&A show that the most complex part of the protocol is G_A, the g-choreography with highlighted background in Figure 5.1. Below we comment on G_A, except for G'_A which is a sequence of interactions immaterial for the rest of the paper.

The g-choreography is a refinement of the CA model shown in Figure 4.1, amended to correct some inaccuracies with the help of one of the key domain experts, OTAM. The first task in the protocol is the nomination of the RCOO, followed by the steps required to enable the RCOO to operate, which

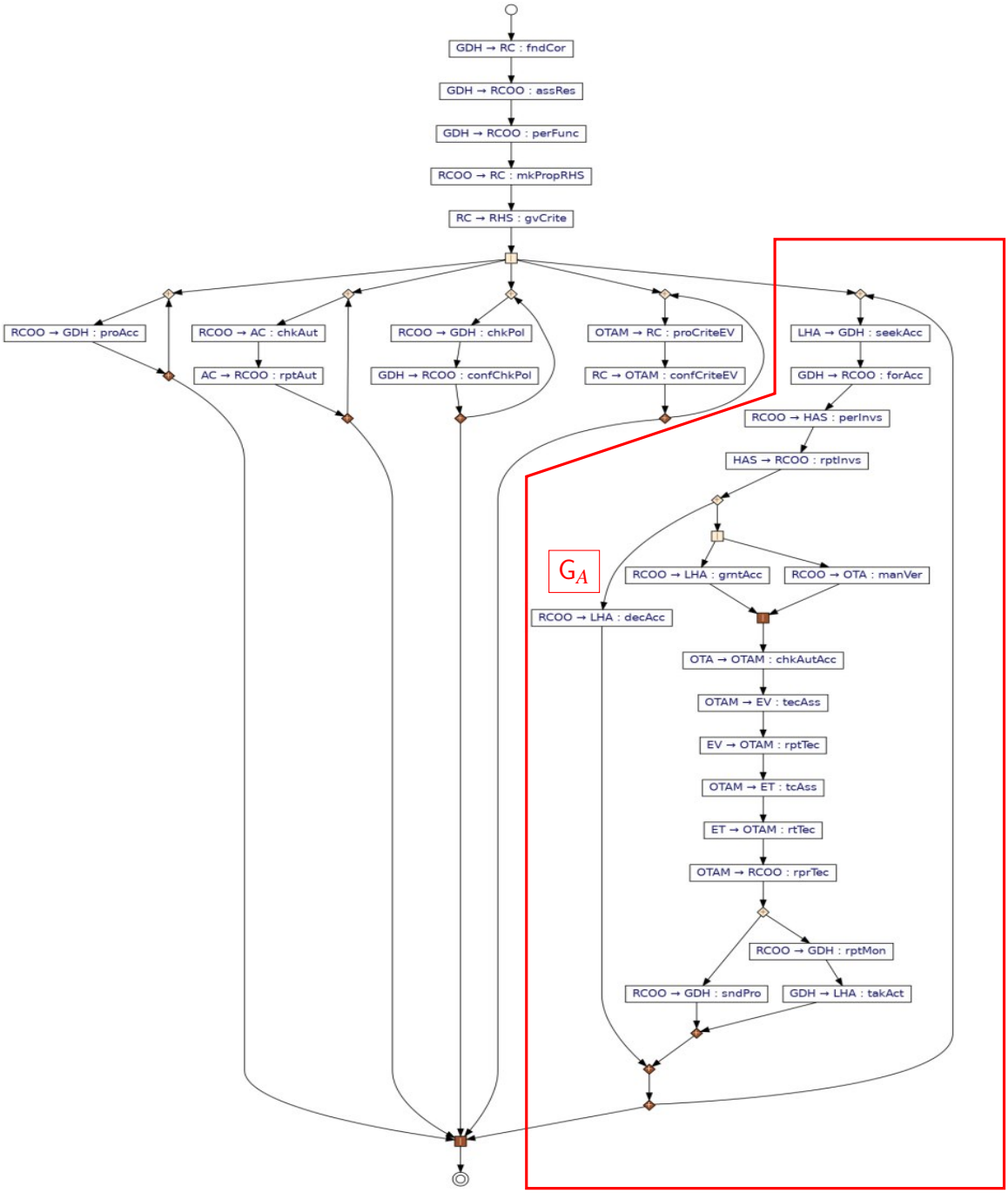


Figure 5.1: G-choreography for Regional Coordination for Authorisation and Accreditation

are the same in both CA and the g-choreography. Afterwards, the protocol describes five concurrent tasks, each of which concerns the iterative execution of a sub-protocol. The concurrent tasks are:

1. The loop at the leftmost branch in the g-choreography corresponds to the self-loop at state 5 in CA;

2. The loop at the second branch from the left in the g-choreography corresponds to the loop involving state 6 in CA;

3. The loop at the third branch from the left in the g-choreography corresponds to the loop involving state 7 in in CA;

4. The loop at the second branch from the right in the g-choreography corresponds to the loop involving state 12 in CA. While modelling CA, we considered the criteria for managing EV as being determined through the coordination between OTAM and RC within the branch where the actual process of accreditation to LHA occurs. However, after consulting with the stakeholders, we found that this represents one of the parallel protocols that takes place after the transition RC→RHS : gvCrite;

5. In the revised version, within the rightmost branch, LHA requests GDH to initiate the actual procedure for accreditation, represented by LHA→GDH : seekAcc. In the previous model of CA, however, RCOO was responsible for requesting OTA to start the procedure. See Section 5.1 for the detailed formalisation of the rightmost branch;

Let us now discuss the protocol formalised by G_A. After the request from LHA to GDH for the initiation of the accreditation process, the request is forwarded by GDH to RCOO (GDH→RCOO : forAcc), which then asks the Hospital Assistance Sector (HAS) to verify whether LHA satisfies the prerequisite criteria for obtaining accreditation (RCOO→HAS : perInvs). In the new protocol, we introduce a new participant, HAS (Italian name is Settore Assistenza Ospedaliera), which was absent in CA. The role of HAS is to conduct an investigation to determine whether LHA meets the basic prerequisite conditions required to either grant or decline accreditation. Hence HAS replies with a report (HAS→RCOO : rptInvs) which triggers a choice:

- either RCOO declines the accreditation request of LHA based on the report sent by the HAS (RCOO→LHA : decAcc);
- or RCOO grants a preliminary accreditation to LHA (RCOO→LHA : grntAcc) and in parallel mandates to the OTA (Technically Accrediting Body) to perform a full verification (RCOO→OTA : manVer) of the LHA organisation. In the previous protocol of CA, the interaction RCOO→OTA : manVer was considered the initial transition of the actual protocol, occurring from state 5 to state 8;

The latter branch above continues as a single flow of execution. After granting accreditation and completing the mandate verification, OTA notifies OTAM to check the authorisation and accreditation status of LHA through the message OTA→OTAM : chkAutAcc. This interaction corresponds to the same process as in the previous model of CA, representing the transition from state 8 to state 9. Thereafter, the transition in the g-choreography corresponds to the transition from state 11 to state 5 in CA.

Although, there are some refinements in the g-choreography, the CA model was also approximating the actual protocol due to the impossibility of handling the combinatorial explosion arising from the necessary explicit enumeration of all interleavings of concurrent activities. This problem was avoided in [47] by turning some concurrent computations into non-deterministic choices. Since such computations were loops, it means that the subprotocols were executed one at the time instead of in interleaving. Besides imprecision, this model yields false positives in the analysis, since non-deterministic composition is subject to stricter conditions than parallel one.

Let us turn our attention to the remarkable structure of G_A. Indeed, G_A is an iterative process where, after the sequence of interactions on top, participants engage in a choice. The right branch is the sequential composition of two

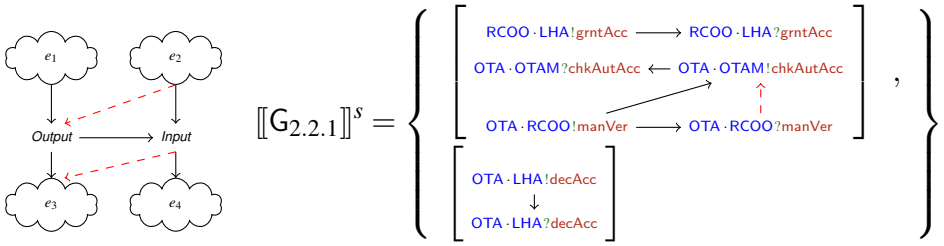
activities in parallel, followed by the interactions in G'_A, and finally a nested choice. As we will see in the rest of the section this is both a source of complexity and the source of subtle glitches in the protocol.

5.2 Synchronous Semantics

In order to cope with different communication models, we extend the framework to support synchronous message passing. The synchronous semantics of a g-choreography G, denoted as $[\![G]\!]^s$, is defined by induction on the syntactic structure of G. The only difference w.r.t. the asynchronous semantics is the case for $G; G'$, which adds the dependencies necessary to model the handshake between sender and receiver: (1) events of a participant in G must precede the events of the same participant in G'; (2) events in G' that depend on an output in G must also depend on the corresponding input in G; (3) events in G preceding an input in G' must also precede the corresponding output in G'.

The synchronous semantics of a choreography can be computed by adding the missing dependencies to its asynchronous semantics: for every pomset and for every output e and its matching successor input e' every predecessor of e' must be a predecessor of e and every successor of e must be successor of e'. Example 5.2.1 below shows a pictorial representation of the translation from asynchronous to synchronous semantics.

Example 5.2.1. *The synchronous semantics of the g-choreography of Example 2.2.1 is* $[\![G_{2.2.1}]\!]^s$*, which is as follows:*



In the synchronous setting, every successor of the event OTA · RCOO!manVer *(here just* OTA · OTAM!chkAutAcc*) must be a successor of* OTA · RCOO?manVer *as well, hence the additional dashed red arrow in picture. The events corresponding to the interaction* OTA→OTAM : chkAutAcc *remain however independent from those events for* RCOO→LHA : grntAcc *since no participant is involved in both the interactions. Moreover, the events for* RCOO→LHA : grntAcc *and the ones for* OTA→RCOO : manVer *are independent of each others since they belong to different threads.* ◇

The choice of modelling interactions also in the synchronous case with two distinct events (send and receive) while preserving independence of concurrent threads allows us to reuse the same theory and machinery of the asynchronous semantics, which is grounded on the work in [35] on Message Sequence Charts. However this results in each interaction (e.g., OTA→RCOO : manVer) being modelled by a two-way handshake (e.g., OTA · RCOO!manVer and OTA · RCOO?manVer) that cannot be interleaved by other messages of the same participants on the same thread (e.g., OTA · OTAM!chkAutAcc), but can be interleaved by other messages on other threads (e.g. RCOO · LHA!grntAcc). An alternative semantics could maintain distinct events per interaction, thus still benefiting from the asynchronous machinery, while also preventing interleaving of other threads between the send and receive events of a single interaction. This would require generating two pomsets per interaction, each enriched with specific dependencies to reflect thread-level ordering. E.g., we would have: one pomset with the additional edge OTA · RCOO?manVer → RCOO · LHA!grntAcc and one pomset with the additional edge RCOO · LHA?grntAcc → RCOO · LHA!manVer.

To avoid introducing new labels, the two-way handshake scheme described above is reflected in the LTS for the synchronous semantics of communicating systems by "overloading" the communication labels to let them represent handshaking. More precisely, the realisation of the synchronous semantics of a g-

choreography involving participants $\mathsf{A}_1,\ldots,\mathsf{A}_n$ can be attained through a *synchronous communicating system*, where a configuration is a tuple of length n whose i-th element is either the current state of (the CFSM of) A_i or the *handshake point* $[q_i l]$ with q_i a state of A_i and l a label on a transition from q_i; call *stable* configurations that do not have handshake points. Let $(q_1,\ldots,q_n)$ be a stable configuration such that $q_i \xrightarrow{\mathsf{A_i}\cdot\mathsf{A_j}!\mathsf{m}} q_i'$ is a transition of A_i and $q_j \xrightarrow{\mathsf{A_i}\cdot\mathsf{A_j}?\mathsf{m}} q_j'$ is a transition of A_j then the synchronous LTS will have the transitions

$$(q_1,\ldots,q_i,\ldots,q_j,\ldots,q_n) \xrightarrow{\mathsf{A_i}\cdot\mathsf{A_j}!\mathsf{m}} (q_1,\ldots,[q_i\mathsf{A_i}\cdot\mathsf{A_j}!\mathsf{m}],\ldots,[q_j\mathsf{A_i}\cdot\mathsf{A_j}?\mathsf{m}],\ldots,q_n) \xrightarrow{\mathsf{A_i}\cdot\mathsf{A_j}?\mathsf{m}} (q_1,\ldots,q_i',\ldots,q_j',\ldots,q_n)$$

(and similarly for the configuration $(q_1,\ldots,q_j,\ldots,q_i,\ldots,q_n)$).

The language of a system of CFSMs is the set of all sequences of labels corresponding to the transitions of runs from the initial configuration.

In order to check realisability in the synchronous setting, we define synchronous well-formedness of pomsets: (i) the pomset is well-formed w.r.t. the asynchronous semantics; (ii) each output event $\mathsf{A}\cdot\mathsf{B}!\mathsf{m}$ has exactly one matching immediate-successor input $\mathsf{A}\cdot\mathsf{B}?\mathsf{m}$. Since the verification conditions over pomsets are parametrised on well-formedness, to check realisability in the synchronous case it is enough to check the closure conditions (cf. Definition 7) using the synchronous well-formedness.

Notice that some choreographies are realisable in the synchronous model while they are not realisable in the asynchronous one. For instance the g-choreography of Example 2.2.1 does not satisfy CC3 in the asynchronous case as seen in Example 2.2.2, but it does in the synchronous semantics. In fact, the counterexample used for the asynchronous case is:
$r = [\mathsf{RCOO}\cdot\mathsf{LHA}!\mathsf{grntAcc} \to \mathsf{RCOO}\cdot\mathsf{LHA}?\mathsf{grntAcc} \quad \mathsf{OTA}\cdot\mathsf{LHA}!\mathsf{decAcc}]$
It is not well-formed in the synchronous model, since the output

$\mathsf{OTA} \cdot \mathsf{LHA}!\mathsf{decAcc}$ has no corresponding input, and therefore it is not considered for the closure condition in the synchronous case. Intuitively, the g-choreography of Example 2.2.1 can be implemented in the synchronous model since OTA and RCOO must synchronise in the right thread of the right branch, making impossible for them to choose two different branches.

Some choreographies are also not realisable in the synchronous model. For instance, replacing the interaction $\mathsf{OTA} \to \mathsf{LHA} : \mathsf{decAcc}$ with $\mathsf{OTAM} \to \mathsf{LHA} : \mathsf{decAcc}$ in the g-choreography in Example 2.2.1 yields the g-choreography $\mathsf{G}'_{5.2}$ with semantics

$$[\![\mathsf{G}'_{5.2}]\!]^s = \left\{ \begin{bmatrix} \mathsf{RCOO} \cdot \mathsf{LHA}!\mathsf{grntAcc} & \longrightarrow & \mathsf{RCOO} \cdot \mathsf{LHA}?\mathsf{grntAcc} \\ \mathsf{OTA} \cdot \mathsf{OTAM}?\mathsf{chkAutAcc} & \longleftarrow & \mathsf{OTA} \cdot \mathsf{OTAM}!\mathsf{chkAutAcc} \\ \mathsf{OTA} \cdot \mathsf{RCOO}!\mathsf{manVer} & \longrightarrow & \mathsf{OTA} \cdot \mathsf{RCOO}?\mathsf{manVer} \end{bmatrix} , \quad \begin{bmatrix} \mathsf{OTAM} \cdot \mathsf{LHA}!\mathsf{decAcc} \\ \downarrow \\ \mathsf{OTAM} \cdot \mathsf{LHA}?\mathsf{decAcc} \end{bmatrix} \right\}$$

where the left pomset is as in $[\![\mathsf{G}_{2.2.1}]\!]^s$, whereas the other consists of $\mathsf{OTAM} \cdot \mathsf{LHA}!\mathsf{decAcc}$ followed by $\mathsf{OTAM} \cdot \mathsf{LHA}?\mathsf{decAcc}$. Since there is no coordination among the participants OTAM and LHA in the latter pomset with the participants OTA and RCOO of the first interaction of a thread of the former pomset, the execution can result in OTAM and LHA taking a branch different than the one taken by OTA and RCOO. Consider the prefixes $\pi(\mathsf{RCOO}) = [\mathsf{OTA} \cdot \mathsf{RCOO}?\mathsf{manVer}]$ and $\pi(\mathsf{OTA}) = [\mathsf{OTA} \cdot \mathsf{RCOO}!\mathsf{manVer}]$ of the projections of the first branch and the projections of the second branch $\pi(\mathsf{LHA}) = [\mathsf{OTAM} \cdot \mathsf{LHA}?\mathsf{decAcc}]$ and $\pi(\mathsf{OTAM}) = [\mathsf{OTAM} \cdot \mathsf{LHA}!\mathsf{decAcc}]$: there is no prefix of R that is at least as permissive as the well-formed pomset in the inter-participant closure of $\bigcup_{\mathsf{A} \in \mathcal{P}} \pi(\mathsf{A})$, namely

$$[\mathsf{OTA} \cdot \mathsf{RCOO}!\mathsf{manVer} \to \mathsf{OTA} \cdot \mathsf{RCOO}?\mathsf{manVer}$$
$$\mathsf{OTAM} \cdot \mathsf{LHA}!\mathsf{decAcc} \to \mathsf{OTAM} \cdot \mathsf{LHA}?\mathsf{decAcc}]$$

5.3 Verifying Realisability

Closure conditions are required to ensure that the behaviour specified by a set of pomsets is preserved under all valid extensions and reorderings consistent with concurrency. They guarantee that the specification is semantically complete and implementable, preventing unrealisable behaviours that arise from missing or inconsistent pomset combinations. The closure conditions must be verified on finite pomsets. Therefore, we consider a bounded unrolling of loops in the implementation, limited to two iterations. This approach allows us to detect coordination issues such as: a message from one loop iteration being confused with a message from the next; participants disagreeing on the number of iterations executed; a message occurring after the loop being mistaken for one within the loop; and events from a loop iteration being reordered with respect to messages in the subsequent iteration. The main source of complexity in verifying closure conditions is that we have to check prefix-language inclusion and in the size of inter-participant closures. The next two sections tackle these problems.

5.3.1 Avoiding pomset compositions explosion

Given a set of pomsets R over participants $\mathcal{P}$, the main challenge in checking CC2 arises from the exponential growth of the number of elements in the set of branch assignments Π_R with the size of $\mathcal{P}$. For example, consider the choreography $\mathsf{G}_{xy} = \mathsf{G}_x + \mathsf{G}_y$, where:

$$\mathsf{G}_x = \mathsf{A}_1 \to \mathsf{A}_2 : x; \mathsf{A}_2 \to \mathsf{A}_3 : x \ldots; \mathsf{A}_{n-1} \to \mathsf{A}_n : x$$
$$\mathsf{G}_y = \mathsf{A}_1 \to \mathsf{A}_2 : y; \mathsf{A}_2 \to \mathsf{A}_3 : y; \ldots; \mathsf{A}_{n-1} \to \mathsf{A}_n : y$$

Here, n participants sequentially exchange either x or y. There are two possible projections per participant, one for each branch, so the set Π_R consists of 2^n

Algorithm 5.1: A old algorithm for CC2

```
CC2(R):
  branches = get_all_branches(R)
  tuples = branches[A_1] × ... × branches[A_n]
  ipc = ∪_{t∈tuples} inter_process_closure(t)
  for r ∈ ipc:
    if not exist_more_permissive(r, R):
      return false
  return true
```

```
get_all_branches(R):
  for A ∈ 𝒫:
    branches[A] = ∅
    for r ∈ R:
      if not exist_more_permissive(r⌊A, branches[A]):
        branches[A] = branches[A] ∪ r⌊A
  return branches
```

combinations.

The pseudocode in Algorithm 5.1 presents an earlier version of the procedure used to verify condition CC2 for a set of pomsets R. The algorithm first computes the projection $r\lfloor_{\mathsf{A}}$ for each participant A in $\mathcal{P}$ and for each pomset r in R, using the function `get_all_branches`. Subsequently, it generates cross products among the projections of different participants to form tuples, and for each tuple t, it constructs the inter-participant closure (*ipc*) by adding the missing transitions between *output* and *input* events. A single tuple may yield multiple *ipc*s, depending on the possible completions of interactions. Consequently, Π_R encompasses all possible combinations of tuples and their corresponding inter-participant closures. The key observation is that most of the pomsets in Π_R are either non-well-formed or share the same largest well-formed prefix relevant for CC3. Moreover, suppose we are given two pomsets r_{A} and r_{B} from the projections of participants $\mathsf{A}, \mathsf{B} \in \mathcal{P}$. If the dependencies in the union of r_{A} and r_{B} cannot be augmented to match an input of B from A, then the well-formedness condition will be violated in every $\pi \in \Pi_R$ containing r_{A} and r_{B}, regardless of the pomsets chosen for the other participants.

Algorithm 5.2: A new algorithm for CC2

```
1  CC2(R, 𝒫):
2      let R_A = {r|_A | r ∈ R} for A ∈ 𝒫
3      choose A ∈ 𝒫
4      (R′, 𝒫′) = (R_A, {A})
5      while 𝒫 \ 𝒫′ ≠ ∅
6          choose A ∈ 𝒫 \ 𝒫′
7          R″ = ∅
8          for r′ ∈ R′, r ∈ R_A :
9              for r″ ∈ □(r ∪ r′):
10                 if ∀A′ · A″?m ∈ r″ such that (A′, A″) ∈ (𝒫′ × {A}) ∪ ({A} × 𝒫′)
11                    ∃ immediate predecessor A′ · A″!m ∈ r″ :
12                    # For synchronous case also check
13                    # ∀A′ · A″!m ∈ r″ ∃ immediate successor A′ · A″?m ∈ r″
14                      R″ = R″ ∪ {r″}
15         (R′, 𝒫′) = (R″, 𝒫′ ∪ {A})
16     return ∀r ∈ R′ ∃r′ ∈ R as permissive as r
```

These observations lead to the new procedure for CC2 in Algorithm 5.2, which checks well-formedness incrementally, pruning invalid partial candidates instead of computing the full Π_R explicitly. Line 4 initialises two variables: variable $\mathcal{P}'$ keeps track of the set of already processed participants, while R' keeps track of the valid partial candidates: pomsets in $\{\Box(\cup_A \pi(A)) \mid \pi \in \Pi_{R\downarrow_{\mathcal{P}'}}\}$ that are well-formed w.r.t. $\mathcal{P}'$.

In practice, the algorithm starts with an initial participant (line 3) and selects the next participant (line 6) as the one that can resolve the largest number of pending input/output dependencies in R'. This heuristic is also applied in the CC3 check, where instead of pruning non-well formed pomsets we keep only the largest well-formed prefixes w.r.t. the processed participants.

When processing the next participant A, the algorithm merges every partial candidate with every projection of A (line 8), iterates over their inter-participant closure (line 9), and adds only the resulting new candidates that satisfy well-formedness w.r.t. all events that involve A (checked on lines 10-11 through a condition that implies well-formedness due the fact that pomsets are obtained

by the inter-participant closure of the projection of the well-formed pomsets of R). Finally, the algorithm verifies the existence of a more permissive pomset in R for every final candidate in R'.

The heuristic above is relevant since the order in which participants are chosen (line 6) significantly affects the efficiency of pruning. For instance, if in G_{xy} above participants are chosen in order $\mathsf{A}_1, \mathsf{A}_2, \ldots, \mathsf{A}_n$ then the size of R' remains 2 throughout the execution. In contrast, a processing order like $\mathsf{A}_1, \mathsf{A}_3, \ldots, \mathsf{A}_{2m+1}, \ldots$ can cause the size of R' to grow up to 2^m pomsets after m iterations, due to the lack of direct interactions between the first m selected participants.

5.3.2 Avoiding prefix explosion

There is a combinatorial blowup to check CC3 due to the examination of all the prefixes of the pomsets in the choreography's semantics, whose number grows exponentially with the number of events across concurrent threads. For example, given the partial order $[e_1 \to e_2 \quad e'_1 \to e'_2]$, there are seven distinct non-empty proper prefixes: $[e_1]$, $[e'_1]$, $[e_1 \quad e'_1]$, $[e_1 \to e_2]$, $[e'_1 \to e'_2]$, $[e_1 \to e_2 \quad e'_1]$, and $[e_1 \quad e'_1 \to e'_2]$. This becomes especially costly in asynchronous semantics, since the sequential composition of interactions targeting different receivers makes the corresponding receive events concurrent.

The pseudocode in Algorithm 5.3 checks condition CC3 for a set of pomsets R. The function get_all_prefixes returns all prefixes of a pomset by simply iterating over all possible subsets of events that satisfy the dependencies. The main observation is that if r is a counterexample (i.e., r is well-formed and no prefix of R is more permissive than r), then every well-formed $r' \in \{\Box(\cup_{\mathsf{A}} \pi(\mathsf{A})) \mid \pi \in \Pi_{R'}\}$ such that r is a prefix of r' is also a counterexam-

Algorithm 5.3: A old algorithm for CC3

```
get_all_prefixes(E, ≤, λ):
  E' = E
  to_process = {∅}
  prefixes = ∅
  while (to_process ≠ ∅):
      prefix = to_process.pop()
      for e ∈ E'\prefix:
          if {e' | e' ≤ e}\prefix = ∅:
            to_process += prefix∪{e}
      prefixes += [prefix, ≤, λ]
  return prefixes


CC3(R):
  branches = get_all_branches(R)
  for A ∈ P:
    prefixes[A] = ∪_{r∈branches[A]} get_all_prefixes(r)
  tuples = prefixes[A1] × ... × prefixes[An]
  ipc = ∪_{t∈tuples} inter_process_closure(t)
  R' = ∪_{r∈R} get_all_prefixes(r)
  for r ∈ ipc:
    if not exist_more_permissive(r, R'):
      return false
  return true
```

ple. Moreover, every pomset admits a unique largest well-formed prefix, which can be obtained by removing all non-well-formed events together with their descendants. Based on this insight, the new approach in Algorithm 5.4 avoids explicitly enumerating all prefixes of R. Instead, it performs the inter-participant closure $R'' = \{\Box(\cup_{\mathsf{A}} \pi(\mathsf{A})) \mid \pi \in \Pi_R\}$, then it computes the largest well-formed prefixes R''' of R'', and finally checks that, for every $r''' \in R'''$, there exists a prefix of R that is more permissive than r'''.

In the actual algorithm, similarly to CC2, we do not explicitly compute the inter-participant closure or the largest well-formed prefix. Instead, we build the largest prefixes incrementally. The variable R' keeps track of the largest prefixes of $\{\Box(\cup_{\mathsf{A}} \pi(\mathsf{A})) \mid \pi \in \Pi_{R\downarrow_{\mathcal{P}'}}\}$ that are well-formed w.r.t. $\mathcal{P}'$. When processing a next participant A, we merge R' with the projections of A and we extend the

Algorithm 5.4: A new algorithm for CC3

```
1  CC3(R, 𝒫):
2      let R_A = {r|_A | r ∈ R} for A ∈ 𝒫
3      choose A ∈ 𝒫
4      (R′, 𝒫′) = (R_A, {A})
5      while 𝒫 \ 𝒫′ ≠ ∅
6          choose A ∈ 𝒫 \ 𝒫′
7          R″ = ∅
8          for r′ ∈ R′, r ∈ R_A:
9              for r″ ∈ □(r ∪ r′):
10                 while ∃e = A′ · A″?m ∈ r″ s.t. (A′, A″) ∈ (𝒫′ × {A}) ∪ ({A} × 𝒫′)
11                         and ∄ immediate predecessor A′ · A″!m ∈ r″:
12                         # For synchronous case also check
13                         # e = A′ · A″!m ∈ r″ and ∄ imm. succ. A′ · A″?m ∈ r″:
14                         r″ = r″ \ subgraph from e
15                 R″ = R″ ∪ {r″}
16         (R′, 𝒫′) = (R″, 𝒫′ ∪ {A})
17     return ∀r ∈ R′ ∃r′ ∈ R as permissive as r
```

candidates with the largest well-formed prefixes. Computing the largest well-formed prefix of a pomset (lines 10-14) is straightforward: we iteratively remove non-well-formed events that involve A along with all events causally dependent on them (i.e., input events without a corresponding immediate predecessor output, and in the synchronous case, output events without a corresponding immediate successor input), until only well-formed events remain.

5.4 Evaluation

We integrated Algorithms 5.2 and 5.4 in a new version of PomCho dubbed PomCho+ [22]. We now describe the application of PomCho+ to A&A and compare it to PomCho.

5.4.1 Verifying the closure conditions on A&A

The application of our analysis to A&A shows that it is realisable neither under the asynchronous nor the synchronous semantics. The main reason for this is the choice at the end of the loop in G_A (cf. Figure 5.1, or the extract A&A below). One branch of the choice contains $\mathsf{RCOO}\to\mathsf{GDH}:\mathsf{sndPro}$, while the other one contains the interactions $\mathsf{RCOO}\to\mathsf{GDH}:\mathsf{rptMon};\mathsf{GDH}\to\mathsf{LHA}:\mathsf{takAct}$. Note that participant LHA is only involved in the latter case, hence if the other branch is taken LHA does not know whether the loop has ended or not. This issue is made worst since LHA is the sender in the first transition of the loop, hence it does not know whether to trigger a new iteration by sending a seekAcc message to GDH or to wait for a message in the current iteration.

The above is the main issue flagged by our analysis, but the analysis in [47] hid it among many false positives given by the treatment of parallel composition and late join (cf. [47, Sections 5-6]). Domain experts confirmed that the CA model in [47] was not faithful to the specification which indeed requires a notification to LHA on both the branches. Adding such notification makes the protocol realisable under both the synchronous and the asynchronous semantics. The realisable protocol is available in Figure 5.2.

The error was induced by the fact that in case of a report (right branch) the LHA needs to react. Hence this communication is more emphasised in the informal description [42] on which our model is based. Even if the analysis has not revealed a real bug in the protocol, it has highlighted an error in the model contributing to its refinement. The absence of false positives was also key to highlight real issues.

We give a snippet $\hat{\mathsf{G}}$ of our model that constitutes a minimal example illustrating the problem described above. This allows us to identify a reasonably small

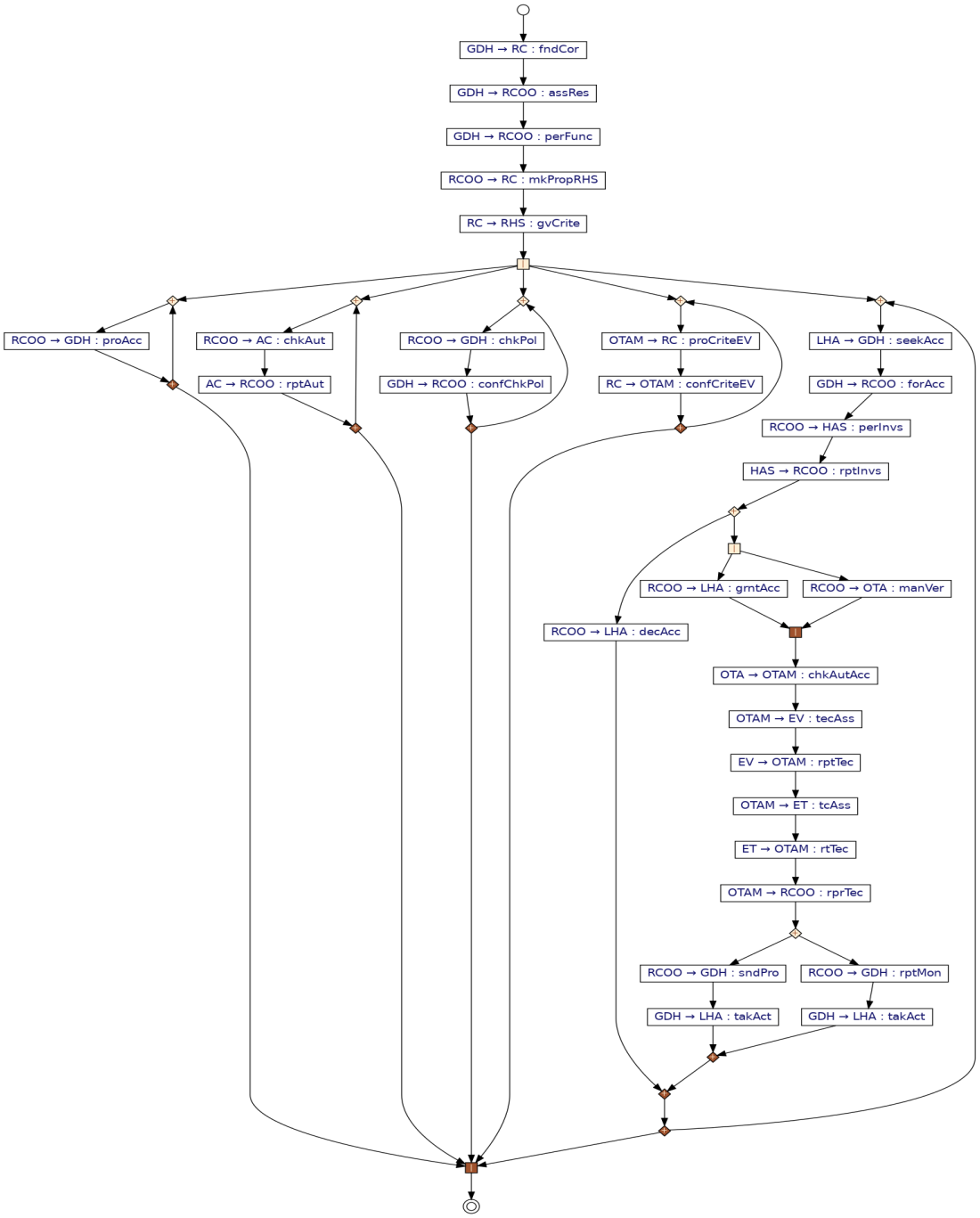
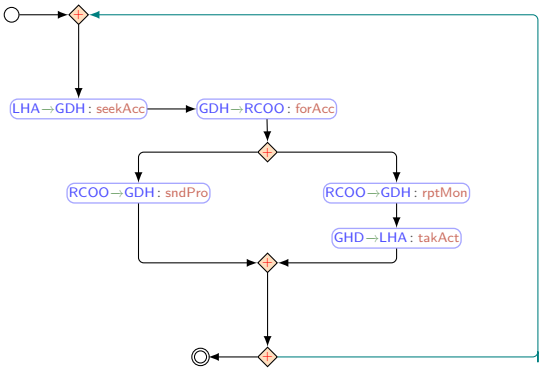


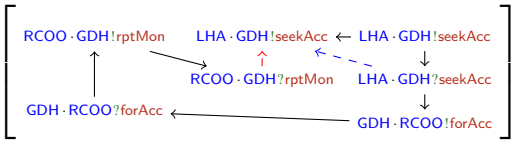
Figure 5.2: Realisable G-choreography of the A&A

pomset flagging the issue instead of analysing the large one corresponding to our full A&A.

$$
\begin{aligned}
\hat{G} = \mathit{repeat}\ & \mathsf{LHA} \to \mathsf{GDH} : \mathsf{seekAcc}; \\
& \mathsf{GDH} \to \mathsf{RCOO} : \mathsf{forAcc}; \\
& (\\
& \quad \mathsf{RCOO} \to \mathsf{GDH} : \mathsf{sndPro} \\
& \quad + \\
& \quad \mathsf{RCOO} \to \mathsf{GDH} : \mathsf{rptMon}; \\
& \quad \mathsf{GHD} \to \mathsf{LHA} : \mathsf{takAct} \\
&)
\end{aligned}
$$



The counterexample of the semantics $[\![\hat{G}]\!]$ and $[\![\hat{G}]\!]^s$ (blue edge is only part of synchronous semantics) of the g-choreography $\hat{G}$ is



The pomset above is constructed from the projection of LHA that corresponds to the left branch for the first loop iteration, while all projections of the other participants correspond to the right branch. This counterexample shows that LHA can initiate the second iteration (i.e., second $\mathsf{LHA} \cdot \mathsf{GDH}!\mathsf{seekAcc}$) without waiting the message takAct from GDH. Therefore, the temporal dependency between $\mathsf{RCOO} \cdot \mathsf{GDH}?\mathsf{rptMon}$ and the second $\mathsf{LHA} \cdot \mathsf{GDH}!\mathsf{seekAcc}$ (i.e., the missing red edge) is violated. For this reason, there is no pomset in $[\![\hat{G}]\!]$ or $[\![\hat{G}]\!]^s$ that has a prefix more permissive than this counterexample.

				CC2			CC3		
		Avg Brc	# WF	$\lvert\Pi\rvert$	Cex	Time (s)	# PR	Cex	Time (s)
$G_{2.2.1}$	AsyncOld	2	16	2	0	0.008	26	3	0.043
	AsyncNew	2	2	2	0	0.006	2	1	0.010
	Sync	2	2	2	0	0.007	2	0	0.009
$\hat{G}$	AsyncOld	6	216	7	1	0.228	56	20	1.091
	AsyncNew	6	12	7	1	0.091	7	3	0.133
	Sync	6	12	7	0	0.130	7	1	0.145
$G_{xy}(n=15)$	AsyncOld	2	32768	2	0	35.392	*	*	*
	AsyncNew	2	2	2	0	0.139	2	0	0.809494
	Sync	2	2	2	0	0.157	2	0	1.165
G_5	AsyncOld	1	1	1	0	0.026	441	0	22.450
	AsyncNew	1	1	1	0	0.024	1	0	0.014
	Sync	1	1	1	0	0.028	1	0	0.029
A&A	AsyncOld	16.18	*	*	*	*	*	*	*
	AsyncNew	16.18	480	224	32	863.790	112	80	439.439
	Sync	16.18	480	224	16	968.946	112	8	445.746
A&A †	AsyncOld	15.64	*	*	*	*	*	*	*
	AsyncNew	15.64	416	192	0	1055.048	80	0	402.141
	Sync	15.64	416	192	0	1048.053	64	0	334.048

Table 5.1: Benchmarks ($\star$ indicates timeout without success after 1800 seconds)

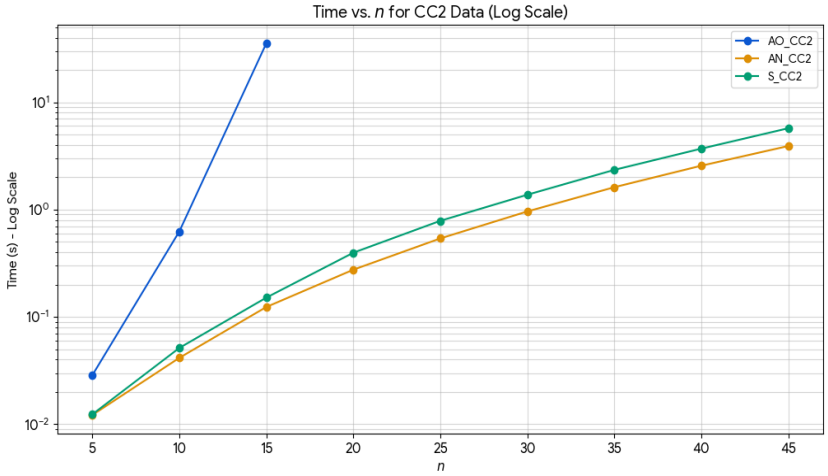
5.4.2 Benchmarking our algorithms

We benchmark our algorithms against those in [48] on examples from previous sections, including A&A. Previous work only supported asynchronous semantics, so we have no synchronous reference. Table 5.1 reports the results of our benchmarks, conducted on a Macbook Air M2, operating system Ubuntu 22.04.5 LTS with 64 bit, Kernel: Linux 5.19.5-10-asahi, arm64 architecture, and 8GB RAM. Each row corresponds to the g-choreography in the first column. The remaining columns report the results on the algorithms considered (sub-rows Old and New for the asynchronous algorithm in [6] and ours, respectively, and Sync for our synchronous algorithm). More precisely, Avg Brc represents the average number of branches per participant; # WF yields the number of analysed combinations of well-formed pomsets; $\lvert\Pi\rvert$ yields the number of pomsets in the inter-participant closure analysed for CC2; # PR yields number of prefixes analysed for CC3; Cex the number of pomsets that do not satisfy the condition for CC2 and CC3 while Time yields the total running time to check CC2 and CC3. Cells with '*' correspond to executions terminated without success after 1800 seconds.

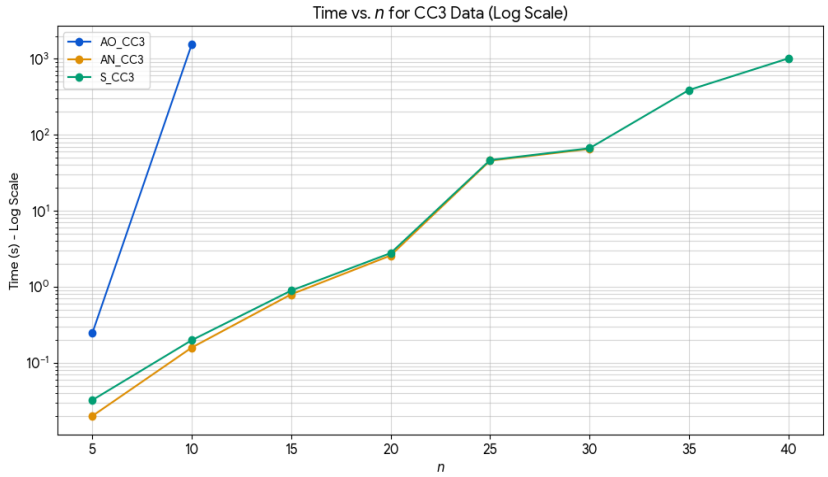
For small examples, such as $\mathsf{G}_{2.2.1}$ of Example 2.2.1 and $\hat{\mathsf{G}}$ of Section 5.4.1, the new algorithms achieve speedups ranging from 1.4× to 8.4×, due to the reduced number of prefixes analysed and the preventive pruning of non-well formed candidates. Notice that $\mathsf{G}_{2.2.1}$ is realisable in the synchronous model only.

We assessed the contribution of the individual optimisations via dedicated examples. The g-choreography G_{xy} of Section 5.3.1 with $n = 15$ allows us to evaluate the impact of the number of participants: PomCho+ achieves a speedup of 272× for CC2 since it maintains only two well-formed pomsets during the whole execution, while the size of Π_R computed with PomCho is $2^{15} = 32768$. It is not possible to compute the speedup for CC3, since the size of the set of branch assignments makes PomCho timeout on CC3. The g-choreography G_5 is a variant of G_{xy} from Section 5.3.1, where in G_x (resp. G_y) participant A sequentially sends messages x_1, x_2, x_3, x_4, and x_5 (resp. y_1, y_2, y_3, y_4, and y_5) to B. As described in Section 5.3.2, the semantics of this choreography consists of one pomset with 20 distinct events with 10 events in each thread. This pomset has 441 prefixes, which have to be computed explicitly by the old approach. The new algorithm is able to maintain only one single largest prefix and achieve a speedup of 1603×.

The old implementation is incapable to analyse A&A. Since there are on average about 16 branches for each of the 11 participants, the size of the set of branch assignments would be $16^{11} \approx 1.7 \times 10^{13}$. Each pomset would also have a large number of prefixes, due to the four threads. In contrast, PomCho+ maintains up to 480 partial well-formed pomsets while analysing CC2, and 112 prefixes while analysing CC3. Similar considerations arise for A&A †, which is A&A updated so to make it realisable.



(a) CC2



(b) CC3

Figure 5.3: G_{xy} **with** $n > 4$ **and** $n < 46$ **with respect to Time (s)**

Figure 5.3 shows the line graph of the execution times of the old and new algorithms for G_{xy} with respect to $n > 4$ and $n < 46$, using a timeout of 1800 seconds. The results are presented for CC2 in Figure 5.3a, and CC3 in Figure 5.3b. *AO* denotes the asynchronous old algorithm, *AN* denotes the asynchronous new algorithm, and *S* denotes the synchronous algorithm. We experimented with different values of n. To check realisability, both CC2 and CC3 conditions must be verified. In the asynchronous model, the old algorithm fails to compute CC2 when $n > 15$ and CC3 when $n > 10$. Therefore, it cannot ver-

ify realisability for $n > 10$, since both conditions must be satisfied. In contrast, the new algorithm can compute CC2 efficiently but fails to compute CC3 when $n > 30$. Because it is difficult to determine the upper bound of n by generating g-choreographies manually, we experimented up to $n = 45$. We observed that the new algorithm computes CC2 easily, but cannot verify realisability when $n > 30$ due to its inability to compute CC3. In the synchronous model, the algorithm fails to compute CC3 when $n > 40$, but it can still compute CC2 up to at least $n = 45$. Consequently, it cannot verify realisability in the synchronous setting for $n > 40$. From this analysis, it is clear that the proposed algorithm outperforms the old algorithm in the asynchronous setting, and that the proposed algorithm in the synchronous setting is also capable of analysing larger interleaving sizes.

5.5 Publication

Pomsets for Process Management: a Healthcare Case Study*

Sourabh Pal[1], Roberto Guanciale[2], Ivan Lanese[1], Emilio Tuosto[3], and Massimo Clo[4]

[1] University of Bologna, Italy; Centre Inria d'Université Côte d'Azur, France
[2] EECS and Digital Futures, KTH Royal Institute of Technology, Sweden
[3] Gran Sasso Science Institute, Italy
[4] Area ICT e Transizione digitale dei servizi al cittadino - Direzione Generale Cura della Persona, Salute e Welfare, Emilia Romagna, Italy

Abstract. Complex coordination protocols are necessary to manage complex organisations. The healthcare management sector is no exception, since different authorities, users, and systems have to interact with each other in order to achieve their organisational goals. In this paper we consider a case study on the authorisation and accreditation of healthcare structures in the Emilia Romagna region in Italy. We specify the case study using *global choreographies* so to enable the analysis of the correctness of its communication patterns using the PomCho tool. This requires to refine PomCho and its underlying theoretical framework. First, we extend PomCho to support not only asynchronous communication, but also synchronous one. Moreover, in both the cases, we provide *a more efficient algorithm* to check closure properties ensuring realisability of choreographies. The new algorithm allows us to check realisability of larger pomsets than before, which makes our approach viable for complex systems such as our case study.

Chapter 6

Related Work

We have used both BPMN collaboration diagrams, CAs and *g-choreographies* as tools for our modelling and analysis. We remark here that BPMN also allows for BPMN choreographies [40], which, like CAs and *g-choreographies*, focus on interactions. However, they do not have a formal semantics, what makes the analysis not grounded on a precise theory. For instance, a single BPMN choreography interaction may represent any number of message passing communications between the same participants, in any order, making it unclear who starts and who ends such an interaction. While BPMN choreographies could be used as a further intermediate step between BPMN collaboration diagrams and CAs or *g-choreographies*, we considered this step not worth the effort. Indeed, we preferred BPMN collaboration diagrams, since they are closer to the natural language specification, making them more suitable than BPMN choreographies as an intermediate step. One could perform verification directly on the BPMN collaboration diagrams, but available techniques are essentially based on model checking. However, model checking suffers from state-space explosion [49, 50], hence also statistical model checking [51], which is a form of simulation, is used. However, the latter cannot provide full guarantees. We instead focus on specific communication properties, and our dedicated techniques are more effective in this specific setting.

Many models focusing on the correctness of message passing in distributed sys-

tems also exist, see, e.g., the survey in [1]. Among the various approaches, conversation protocols [28], which are non-deterministic Büchi automata whose labels resemble our interactions, are one of the closest to CAs. Also conversation protocols require *well-formedness* conditions, however the comparison between the two sets of *well-formedness* conditions is not easy, as also mentioned in [14]. However, broadly speaking, the conditions on *conversation protocols* are more global, hence when they are not satisfied it is not easy to get feedback about which part of the automaton makes the condition fail. In CAs, feedback is instead local and one can exploit it to update the protocol to fix the issue.

DCR graphs [52, 53] are also a model to represent and analyse distributed systems while avoiding explosion of the representation due to concurrency, but they focus on events and dependencies among them instead of on communications, hence they are less suitable to analyse properties related to message passing.

The concept of selective participation studied in [20] is closely aligned with the approaches in [54] and [55], although they are presented using different formalisms. In [54], causal-based projection analyses causal dependencies between messages so that a participant's local type includes only those interactions that are causally relevant to it, effectively allowing participants to ignore unrelated parts of the protocol. In contrast, [55] relaxes the knowledge-of-choice requirement, so that participants are not forced to be informed of choices or loops when those decisions do not affect their behaviour.

The dynamic creation of participants remains challenging for choreography-based formalisms that assume a fixed set of roles and focus on global specifications amenable to formal verification. In the context of choreographic programming, dynamic process creation has been addressed through mechanisms such as local spawning, service invocation, and subsequent connection of newly created processes [56–58]. These approaches are primarily programming-oriented and rely on runtime constructs and assumptions that differ from those underlying

the global, verification-oriented choreography models considered in this thesis.

Beyond parallel composition, other features of protocols have been studied in the literature, such as having a variable number of participants play a given role in the choreography [59] (multirole), or the possibility of having protocols where the number of some messages is parametric on a previously transmitted integer [60]. While the latter behavior never occurred in A&A, the former could be used to better model roles such as EVs or ETs. However, in A&A EVs or ETs always act as a single participant, hence it is not clear whether the added complexity of multiroles provides any more insight on the correctness of the protocol. This question is left for future work.

While in the research we have not stressed the projection from CA to local descriptions so to enable a correct by construction implementation, we recall here that [14] proves that a simple homomorphic projection allows one to generate such local descriptions. However, as showed in [61], there are cases where a correct implementation exists, but in order to generate it more refined techniques are needed. Such cases never occur in A&A, hence the simple projection in [14] would be fine, once the *well-formedness* issues discussed in Section 4.2 are solved.

As mentioned in the discussion, it would be good to extend CAs with support for parallel composition. Parallel composition was supported originally in multiparty asynchronous global types [7,62], but these parallel branches were required to have disjoint sets of participants. This is not enough in A&A, where for instance we would like to specify that the RCOO participates to different subprotocols in parallel. Also, multiparty session types (see, e.g., the survey in [1]) need to satisfy many syntactic conditions on the protocol. While these conditions are useful to enforce good communication properties, they make it impossible to model not well-behaved systems. This is an issue in our case, where we want to get feedback on existing, possibly not well-behaved, proto-

cols.

In [63] a mechanism to statically detect realisability in MSCs (a formal model of *global specifications*) has been proposed; the analysis is based on notions of non-local choices and of termination that are less permissive than our verification conditions since intra-participant concurrency is not allowed and termination awareness is not enforced.

Realisation of *global views* allowing interactions between multiple *senders* and multiple *receivers* is considered in [64]. Two notions of *realisability* are discussed, depending on whether local actions include the name of involved participants or not. Tool support is provided by the Ceta [65] tool.

Partial order models of choreographies akin to pomsets have been recently proposed in [66–68]. The pomset semantics of *g-choreographies* in [23, 24] has inspired *branching pomsets* [67,68], which enable a more compact representation of choices. Whether this representation offers better algorithms for checking closure conditions is an open problem. Similar considerations apply to the results in [69], which proposes an optimisation of the pomset representation of concurrent computations.

The platform in [70] offers a framework for the top-down development of message-passing applications with low-level features (e.g., binding of components, or security aspects) but has limited functionalities for the analysis of *global specifications*.

Chapter 7

Conclusion and Future Work

We have modelled the protocol for *accreditation and authorisation* of healthcare structures in the Emilia Romagna region, Italy, using first a BPMN collaboration diagram and then a CA and *g-choreography*. We used the tool Corinne to check the *well-formedness* properties of the resulting CA. We also used the tool PomCho+ to check the realisability of the *g-choreography*.

As far as the use of CAs is concerned, the analysis gave interesting feedback on how to extend Corinne and the underlying theory to better support the analysis of such a kind of protocols, as discussed in Chapter 4. While not highlighting any real issue in the protocol, the modelling exercise has been useful to give a more precise specification of the protocol, and highlighting its message passing aspect.

The analysis on the case study in Chapter 4 uses CAs and the Corinne tool. A main drawback of CAs (and, consequently, of Corinne) is that concurrency is modelled by explicitly representing all interleavings. This makes both the diagrammatic representation and the analysis quickly unfeasible for highly concurrent cases, such as Figure 4.1. The problem was avoided in Chapter 4 by turning some concurrent computations into non-deterministic choices. This approach has two drawbacks; firstly it does not faithfully model Figure 4.1, secondly, it produces false positives since non-deterministic choice is subject to

stricter conditions than parallel composition.

Parallel composition, where the same participant can act in multiple parallel branches, can be specified, for instance, in g-choreographies [6, 24]. The conditions in [6] are close to those of *conversation protocols*, while those in [24] are closer to CAs. Analysis can be automated using the PomCho tool suite [23]. Therefore, we adopt *g-choreography* for further analysis and to handle concurrency. However, the existing PomCho was built for *asynchronous communication*, whereas our work is based on *synchronous communication* using CAs. We believe that *synchronous communication* sometimes better captures human-based synchronisation. Moreover, the existing implementation of the *asynchronous* model was unable to analyse large-scale case studies like our. To address this limitation, we extended PomCho with new algorithms supporting both *asynchronous* and *synchronous* semantics, and we refer to the enhanced version as PomCho+. We then analysed the A&A using both *asynchronous* and *synchronous semantics*.

Hence, to provide a satisfactory analysis of A&A, we turned to g-choreographies that compactly capture parallelism. This is convenient also because computations of g-choreographies can be compactly expressed as pomsets, which are the structures used in the tool PomCho [23] to check the closure conditions ensuring *well-formedness* of g-choreographies. However, A&A also challenges PomCho since the check of the *closure conditions* is implemented as a brute force algorithm. To overcome this issue we introduced optimisations of the algorithms and implemented them in PomCho+. The verification of A&A initiates the usability analysis of PomCho advocated in [23].

The *closure conditions* proposed in [24] take inspiration from those in [35] and allow a more efficient analysis [23]. We extend the approach in [24] to *synchronous* interactions and adapt PomCho+ to support the new *closure conditions*. These results are instrumental to tackle the complexity of our case study.

The theories underlying PomCho and Corinne have common aims but different formalisations; we plan to investigate their relation in future work.

Future Work. The main motivation for our work is to verify whether CAs and *g-choreographies* are suitable to analyse scenarios from healthcare management, and whether CAs and *g-choreographies* can bring added value to healthcare management. While initial results are promising, as we discussed in Chapter 4, we believe the current theory of CAs and the Corinne tool need to be improved to support *late join* and dynamic creation of participants to better analyse this kind of case studies. Also, right now *well-formedness* checks of Corinne are assuming a *synchronous* communication framework. This has not emerged as a main limitation for A&A, where communications are mostly human-based, however adding checks for an *asynchronous* implementation [14] would be interesting, to pave the way to the use of the *projection* operation of CAs (which generates *Communicating Finite State Machines* [13]) to generate *correct-by-construction* support to execute the protocol.

Beyond working on those improvements, and considering additional case studies, we plan to improve on the feedback provided by Corinne. Indeed, Corinne raises many issues which are similar as distinct errors, e.g., issues of *late join* related to a same loop and many other paths in the CA. While adding support for *late join* would at least mitigate this issue, it would be interesting to automatically group highlighted issues which rely on the same or similar causes. This would allow to provide better feedback to the user.

As far as *g-choreography* is concerned, the *synchronous semantics* and analysis build on the *asynchronous* ones. The reason was to reuse as much as possible previous PomCho implementation. One could instead represent *synchronous* communication as a single interaction event, labelled $A \rightarrow B : m$. This enables to halve the number of events (obtaining a corresponding speedup), but requires an extensive rework of both the tool and the theory.

The current integrated approach could also help to exploit both the *synchronisation* mechanisms in a same model. This would allow to study heterogeneous systems.

References

[1] Hans Hüttel, Ivan Lanese, Vasco T. Vasconcelos, Luís Caires, Marco Carbone, Pierre-Malo Deniélou, Dimitris Mostrous, Luca Padovani, António Ravara, Emilio Tuosto, Hugo Torres Vieira, and Gianluigi Zavattaro. Foundations of session types and behavioural contracts. *ACM Comput. Surv.*, 49(1):3:1–3:36, 2016.

[2] Alex Coto, Franco Barbanera, Ivan Lanese, Davide Rossi, and Emilio Tuosto. On formal choreographic modelling: A case study in EU business processes. In Tiziana Margaria and Bernhard Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation. Verification Principles - 11th International Symposium, ISoLA 2022, Rhodes, Greece, October 22-30, 2022, Proceedings, Part I*, volume 13701 of *Lecture Notes in Computer Science*, pages 205–219. Springer, 2022.

[3] Mario Bravetti, Marco Carbone, Thomas T. Hildebrandt, Ivan Lanese, Jacopo Mauro, Jorge A. Pérez, and Gianluigi Zavattaro. Towards global and local types for adaptation. In Steve Counsell and Manuel Núñez, editors, *Software Engineering and Formal Methods - SEFM 2013 Collocated Workshops: BEAT2, WS-FMDS, FM-RAIL-Bok, MoKMaSD, and OpenCert, Madrid, Spain, September 23-24, 2013, Revised Selected Papers*, volume 8368 of *Lecture Notes in Computer Science*, pages 3–14. Springer, 2013.

[4] OMG. Business Process Model and Notation (BPMN), Version 2.0, January 2011. `https://www.omg.org/spec/BPMN`.

[5] Jonas Bonér. *Reactive Microsystems: The Evolution of Microservices at Scale*. O'Reilly Media, 2017.

[6] Emilio Tuosto and Roberto Guanciale. Semantics of global view of choreographies. *J. Log. Algebraic Methods Program.*, 95:17–40, 2018.

[7] Kohei Honda, Nobuko Yoshida, and Marco Carbone. Multiparty asynchronous session types. *Journal of the ACM (JACM)*, 63(1):1–67, 2016.

[8] Mario Coppo, Mariangiola Dezani-Ciancaglini, Nobuko Yoshida, and Luca Padovani. Global progress for dynamically interleaved multiparty sessions. *Mathematical Structures in Computer Science*, 26(2):238–302, 2016.

[9] Alceste Scalas and Nobuko Yoshida. Less is more: multiparty session types revisited. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–29, 2019.

[10] Ajay Krishna, Pascal Poizat, and Gwen Salaün. Checking business process evolution. *Science of Computer Programming*, 170:1–26, 2019.

[11] Ivan Lanese, Claudio Guidi, Fabrizio Montesi, and Gianluigi Zavattaro. Bridging the gap between interaction-and process-oriented choreographies. In *2008 Sixth IEEE International Conference on Software Engineering and Formal Methods*, pages 323–332. IEEE, 2008.

[12] Fabrizio Montesi. *Choreographic Programming*. PhD thesis, IT University of Copenhagen, Copenhagen, Denmark, 2013.

[13] Daniel Brand and Pitro Zafiropulo. On Communicating Finite-State Machines. *Journal of the ACM*, 30(2):323–342, 1983.

[14] Franco Barbanera, Ivan Lanese, and Emilio Tuosto. Choreography automata. In Simon Bliudze and Laura Bocchi, editors, *Coordination Models and Languages - 22nd IFIP WG 6.1 International Conference, COORDINATION 2020, Held as Part of the 15th International Federated Conference on Distributed*

Computing Techniques, DisCoTec 2020, Valletta, Malta, June 15-19, 2020, Proceedings, volume 12134 of *Lecture Notes in Computer Science*, pages 86–106. Springer, 2020.

[15] Roberto Guanciale and Emilio Tuosto. An abstract semantics of the global view of choreographies. In Massimo Bartoletti, Ludovic Henrio, Sophia Knight, and Hugo Torres Vieira, editors, *Proceedings 9th Interaction and Concurrency Experience, ICE 2016, Heraklion, Greece, 8-9 June 2016*, volume 223 of *EPTCS*, pages 67–82, 2016.

[16] Joseph Barjis. The importance of business process modeling in software systems design. *Sci. Comput. Program.*, 71(1):73–87, 2008.

[17] Simone Orlando, Vairo Di Pasquale, Franco Barbanera, Ivan Lanese, and Emilio Tuosto. Corinne, a tool for choreography automata. In Gwen Salaün and Anton Wijs, editors, *Formal Aspects of Component Software - 17th International Conference, FACS 2021, Virtual Event, October 28-29, 2021, Proceedings*, volume 13077 of *Lecture Notes in Computer Science*, pages 82–92. Springer, 2021.

[18] Sourabh Pal, Ivan Lanese, and Emilio Tuosto. Corinne-3. Available at `https://github.com/lanese/corinne-3`.

[19] DOT. Available at `https://graphviz.org/doc/info/lang.html`.

[20] Lorenzo Gheri, Ivan Lanese, Neil Sayers, Emilio Tuosto, and Nobuko Yoshida. Design-by-contract for flexible multiparty session protocols. In Karim Ali and Jan Vitek, editors, *36th European Conference on Object-Oriented Programming, ECOOP 2022, June 6-10, 2022, Berlin, Germany*, volume 222 of *LIPIcs*, pages 8:1–8:28. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[21] PomCho Tool repository. Available at `https://bitbucket.org/sourabhphd/chorgram/src/workingcoppy/`.

[22] PomCho+ Tool repository. Available at https://bitbucket.org/sourabhphd/chorgram/src/ICTAC_2025/.

[23] Roberto Guanciale and Emilio Tuosto. Pomcho: A tool chain for choreographic design. *Sci. Comput. Program.*, 202:102535, 2021.

[24] Roberto Guanciale and Emilio Tuosto. Realisability of pomsets. *J. Log. Algebraic Methods Program.*, 108:69–89, 2019.

[25] Formal description techniques (FDT) - Message Sequence Chart (MSC). Recommendation ITU-T Z.120, 2011. Available at http://www.itu.int/rec/T-REC-Z.120-201102-I/en.

[26] Emmanuel Gaudin and Eric Brunel. Property Verification with MSC. In *International SDL Forum*, Lecture Notes in Computer Science, pages 19–35. Springer, 2013.

[27] Object Management Group. Business Process Model and Notation, 2011. http://www.bpmn.org.

[28] Xiang Fu, Tevfik Bultan, and Jianwen Su. Conversation protocols: a formalism for specification and verification of reactive electronic services. *Theor. Comput. Sci.*, 328(1-2):19–37, 2004.

[29] Mario Bravetti and Gianluigi Zavattaro. Towards a unifying theory for choreography conformance and contract compliance. In *International Conference on Software Composition*, pages 34–50. Springer, 2007.

[30] Marco Carbone, Kohei Honda, and Nobuko Yoshida. Structured communication-centred programming for web services. In *European Symposium on Programming*, pages 2–17. Springer, 2007.

[31] Adrian Francalanza, Claudio Antares Mezzina, and Emilio Tuosto. Reversible choreographies via monitoring in erlang. In *IFIP International Conference on Distributed Applications and Interoperable Systems*, pages 75–92. Springer, 2018.

[32] Mila Dalla Preda, Maurizio Gabbrielli, Saverio Giallorenzo, Ivan Lanese, and Jacopo Mauro. Dynamic choreographies: Theory and implementation. *Logical Methods in Computer Science*, 13, 2017.

[33] Ivan Lanese, Fabrizio Montesi, and Gianluigi Zavattaro. Amending choreographies. In António Ravara and Josep Silva, editors, *Proceedings 9th International Workshop on Automated Specification and Verification of Web Systems, WWV 2013, Florence, Italy, 6th June 2013*, volume 123 of *EPTCS*, pages 34–48, 2013.

[34] Vaughan Pratt. Modeling concurrency with partial orders. *International journal of parallel programming*, 15:33–71, 1986.

[35] Rajeev Alur, Kousha Etessami, and Mihalis Yannakakis. Inference of Message Sequence Charts. *IEEE Trans. Software Eng.*, 29(7):623–633, 2003.

[36] PomCho+ Tool repository: An example that satisfies CC3 but not CC2. Available at `https://bitbucket.org/sourabhphd/chorgram/src/ICTAC_2025/ICTAC_Examples/CC2CC3/`.

[37] Nickolas Kavantzas, David Burdett, Gregory Ritzinger, Tony Fletcher, Yves Lafon, and Charlton Barreto. Web services choreography description language version 1.0. *W3C candidate recommendation*, 9:290–313, 2005.

[38] Marco Autili, Paola Inverardi, and Massimo Tivoli. Automated synthesis of service choreographies. *IEEE Softw.*, 32(1):50–57, 2015.

[39] Kohei Honda, Nobuko Yoshida, and Marco Carbone. Multiparty asynchronous session types. *J. ACM*, 63(1):9:1–9:67, 2016.

[40] BPMN choreography diagrams documentation. Available at `https://www.ibm.com/docs/en/rational-soft-arch/9.7.0?topic=diagrams-bpmn-choreography`.

[41] Healthcare Authorization and Accreditation protocol. Available at `https://salute.regione.emilia-romagna.it/ssr/`

strumenti-e-informazioni/autorizzazione-e-accreditamento/autorizzazione-e-accreditamento-sanitario.

[42] Legge Regionale 06 Novembre 2019. Available at https://demetra.regione.emilia-romagna.it/al/articolo?urn=er:assemblealegislativa:legge:2019;22.

[43] Sourabh Pal, Ivan Lanese, and Massimo Clo. BPMN collaboration diagram of the Regional Coordination for Healthcare Authorization and Accreditation protocol. Available at https://drive.google.com/file/d/1c_Y78dTc8Qh8iXjky6nW71NTbp-HrKdz/view?usp=sharing.

[44] Greta Adamo, Stefano Borgo, Chiara Di Francescomarino, Chiara Ghidini, Nicola Guarino, and Emilio M. Sanfilippo. Business processes and their participants: An ontological perspective. In Floriana Esposito, Roberto Basili, Stefano Ferilli, and Francesca A. Lisi, editors, *AI*IA 2017 Advances in Artificial Intelligence - XVIth International Conference of the Italian Association for Artificial Intelligence, Bari, Italy, November 14-17, 2017, Proceedings*, volume 10640 of *Lecture Notes in Computer Science*, pages 215–228. Springer, 2017.

[45] Raymond Hu and Nobuko Yoshida. Explicit connection actions in multiparty session types. In Marieke Huisman and Julia Rubin, editors, *Fundamental Approaches to Software Engineering - 20th International Conference, FASE 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings*, volume 10202 of *Lecture Notes in Computer Science*, pages 116–133. Springer, 2017.

[46] Paul Harvey, Simon Fowler, Ornela Dardha, and Simon J. Gay. Multiparty session types for safe runtime adaptation in an actor language. In Anders Møller and Manu Sridharan, editors, *35th European Conference on Object-Oriented Programming, ECOOP 2021, July 11-17, 2021, Aarhus, Den-*

mark (Virtual Conference), volume 194 of *LIPIcs*, pages 10:1–10:30. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[47] Sourabh Pal, Ivan Lanese, and Massimo Clo. Choreographic automata: A case study in healthcare management. In *COORDINATION*, volume 14676 of *Lecture Notes in Computer Science*, pages 3–19. Springer, 2024.

[48] Roberto Guanciale and Emilio Tuosto. Realisability of pomsets via communicating automata. In *Proceedings 11th Interaction and Concurrency Experience, ICE*, pages 37–51, 2018.

[49] Flavio Corradini, Fabrizio Fornari, Andrea Polini, Barbara Re, Francesco Tiezzi, and Andrea Vandin. A formal approach for the analysis of BPMN collaboration models. *J. Syst. Softw.*, 180:111007, 2021.

[50] Xiang Fu, Tevfik Bultan, and Jianwen Su. Analysis of interacting BPEL web services. In Stuart I. Feldman, Mike Uretsky, Marc Najork, and Craig E. Wills, editors, *Proceedings of the 13th international conference on World Wide Web, WWW 2004, New York, NY, USA, May 17-20, 2004*, pages 621–630. ACM, 2004.

[51] Koushik Sen, Mahesh Viswanathan, and Gul Agha. Statistical model checking of black-box probabilistic systems. In *International Conference on Computer Aided Verification*, pages 202–215. Springer, 2004.

[52] Søren Debois, Thomas T. Hildebrandt, and Tijs Slaats. Replication, refinement & reachability: complexity in dynamic condition-response graphs. *Acta Informatica*, 55(6):489–520, 2018.

[53] Thomas T. Hildebrandt, Tijs Slaats, Hugo A. López, Søren Debois, and Marco Carbone. Declarative choreographies and liveness. In Jorge A. Pérez and Nobuko Yoshida, editors, *Formal Techniques for Distributed Objects, Components, and Systems - 39th IFIP WG 6.1 International Conference, FORTE 2019, Held as Part of the 14th International Federated Conference on Distributed Computing Techniques, DisCoTec 2019, Kongens Lyngby, Denmark,*

June 17-21, 2019, Proceedings, volume 11535 of *Lecture Notes in Computer Science*, pages 129–147. Springer, 2019.

[54] Rupak Majumdar, Madhavan Mukund, Felix Stutz, and Damien Zufferey. Generalising projection in asynchronous multiparty session types. In Serge Haddad and Daniele Varacca, editors, *32nd International Conference on Concurrency Theory, CONCUR 2021, Virtual Conference, August 24-27, 2021*, volume 203 of *LIPIcs*, pages 35:1–35:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[55] Luís Cruz-Filipe, Fabrizio Montesi, and Robert R Rasmussen. Keep me out of the loop: a more flexible choreographic projection. In *LPAR*, pages 144–163, 2023.

[56] Marco Carbone and Fabrizio Montesi. Deadlock-freedom-by-design: multiparty asynchronous global programming. *ACM SIGPLAN Notices*, 48(1):263–274, 2013.

[57] Bjørn Angel Kjer, Luís Cruz-Filipe, and Fabrizio Montesi. From infinity to choreographies: Extraction for unbounded systems. In *International Symposium on Logic-Based Program Synthesis and Transformation*, pages 103–120. Springer, 2022.

[58] Luís Cruz-Filipe and Fabrizio Montesi. Procedural choreographic programming. In *International Conference on Formal Techniques for Distributed Objects, Components, and Systems*, pages 92–107. Springer, 2017.

[59] Pierre-Malo Deniélou and Nobuko Yoshida. Dynamic multirole session types. In Thomas Ball and Mooly Sagiv, editors, *Proceedings of the 38th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2011, Austin, TX, USA, January 26-28, 2011*, pages 435–446. ACM, 2011.

[60] Peter Thiemann and Vasco T. Vasconcelos. Label-dependent session types. *Proc. ACM Program. Lang.*, 4(POPL):67:1–67:29, 2020.

[61] Elaine Li, Felix Stutz, Thomas Wies, and Damien Zufferey. Complete multiparty session type projection with automata. In Constantin Enea and Akash Lal, editors, *Computer Aided Verification - 35th International Conference, CAV 2023, Paris, France, July 17-22, 2023, Proceedings, Part III*, volume 13966 of *Lecture Notes in Computer Science*, pages 350–373. Springer, 2023.

[62] Kohei Honda, Nobuko Yoshida, and Marco Carbone. Multiparty asynchronous session types. In George C. Necula and Philip Wadler, editors, *Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2008, San Francisco, California, USA, January 7-12, 2008*, pages 273–284. ACM, 2008.

[63] Hanêne Ben-Abdallah and Stefan Leue. Syntactic detection of process divergence and non-local choice in message sequence charts. In Ed Brinksma, editor, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 259–274, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[64] Maurice H ter Beek, Rolf Hennicker, and José Proença. Realisability of global models of interaction. In *International Colloquium on Theoretical Aspects of Computing*, pages 236–255. Springer, 2023.

[65] Ceta Tool. Available at `https://github.com/arcalab/choreo/tree/ceta`.

[66] Ilaria Castellani, Mariangiola Dezani-Ciancaglini, and Paola Giannini. Event structure semantics for multiparty sessions. In *Models, Languages, and Tools for Concurrent and Distributed Programming - Essays Dedicated to Rocco De Nicola on the Occasion of His 65th Birthday*, volume 11665 of *Lecture Notes in Computer Science*, pages 340–363. Springer, 2019.

[67] Luc Edixhoven, Sung-Shik Jongmans, José Proença, and Ilaria Castellani. Branching pomsets: Design, expressiveness and applications to choreographies. *J. Log. Algebraic Methods Program.*, 136:100919, 2024.

[68] Luc Edixhoven and Sung-Shik Jongmans. Realisability of branching pomsets. In Silvia Lizeth Tapia Tarifa and José Proença, editors, *Formal Aspects of Component Software - 18th International Conference, FACS 2022, Virtual Event, November 10-11, 2022, Proceedings*, volume 13712 of *Lecture Notes in Computer Science*, pages 185–204. Springer, 2022.

[69] Luc Edixhoven. Shuffling posets on trajectories (technical report). *CoRR*, abs/2309.09189, 2023.

[70] Marco Autili, Amleto Di Salle, Francesco Gallo, Claudio Pompilio, and Massimo Tivoli. Chorevolution: Automating the realization of highly-collaborative distributed applications. In *Coordination Models and Languages*, pages 92–108. Springer, 2019.