



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

ARCHIVIO ISTITUZIONALE DELLA RICERCA

Alma Mater Studiorum Università di Bologna Archivio istituzionale della ricerca

Formal Choreographic Languages

This is the final peer-reviewed author's accepted manuscript (postprint) of the following publication:

Published Version:

Formal Choreographic Languages / Barbanera F.; Lanese I.; Tuosto E.. - STAMPA. - 13271:(2022), pp. 121-139. (Intervento presentato al convegno COORDINATION tenutosi a Lucca nel 14-16/6/2022) [10.1007/978-3-031-08143-9_8].

Availability:

This version is available at: <https://hdl.handle.net/11585/904285> since: 2022-11-19

Published:

DOI: http://doi.org/10.1007/978-3-031-08143-9_8

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (<https://cris.unibo.it/>).
When citing, please refer to the published version.

(Article begins on next page)

This is the final peer-reviewed accepted manuscript of:

Barbanera, F., Lanese, I., Tuosto, E. (2022). Formal Choreographic Languages. In: ter Beek, M.H., Sirjani, M. (eds) Coordination Models and Languages. COORDINATION 2022. IFIP Advances in Information and Communication Technology, vol 13271. Springer, Cham.

The final published version is available online at: https://doi.org/10.1007/978-3-031-08143-9_8

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (<https://cris.unibo.it/>)

When citing, please refer to the published version.

Formal Choreographic Languages [★]

Franco Barbanera¹, Ivan Lanese², and Emilio Tuosto³
barba@dmi.unict.it, ivan.lanese@gmail.com, emilio.tuosto@gssi.it

¹ Dept. of Mathematics and Computer Science, University of Catania (Italy)

² Focus Team, University of Bologna/INRIA (Italy)

³ Gran Sasso Science Institute (Italy)

Abstract. We introduce a meta-model based on formal languages, dubbed *formal choreographic languages*, to study message-passing systems. Our main motivation is to establish a framework for the comparison and generalisation of standard constructions and properties from the literature. In particular, we consider notions such as global view, local view, and projections from the former to the latter. The correctness of local views projected from global views is characterised in terms of a closure property. A condition is also devised to guarantee relevant communication properties such as (dead)lock-freedom. Formal choreographic languages capture existing formalisms for message-passing systems; we detail the cases of multiparty session types and choreography automata. Unlike many other models, formal choreographic languages can naturally model systems exhibiting non-regular behaviour.

1 Introduction

Choreographic models of message-passing systems are gaining momentum both in academia [8,12,13] and industry [27,34,10]. These models envisage the so-called *global* and *local* views of communicating systems. The former can be thought of as holistic descriptions of protocols that a number of participants should realise through some communication actions, the latter as descriptions of the contribution of single participants.

We propose *formal choreographic languages* (FCL) as a general framework to formalise message-passing systems; existing choreographic models can be conceived as specifications of FCLs. Specifically, we introduce *global* and *local* languages. Global languages (g-languages for short) are made of words built out of

[★] Research partly supported by the EU H2020 RISE programme under the Marie Skłodowska-Curie grant agreement No 778233. Work partially funded by MIUR project PRIN 2017FTXR7S *IT MATTERS* (Methods and Tools for Trustworthy Smart Systems). The first and second authors have also been partially supported by INdAM as members of GNCS (Gruppo Nazionale per il Calcolo Scientifico). The first author has also been partially supported by Progetto di Ateneo UNICT PIACERI. The authors thank the anonymous reviewers for their helpful comments, in particular one reviewer of a previous submission for suggesting the relation with Galois connections. The authors also thank Mariangiola Dezani-Ciancaglini for her support.

interactions of the form $A \rightarrow B:m$, representing the fact that participant A sends message m to participant B , and participant B receives it. Local languages (l-languages for short) consist of words of *actions* of the forms $AB?m$ and $AB!m$, respectively representing that participant B receives message m from A and that participant A sends message m to B .

Abstractly such languages consist of runs of a system described in terms of sequences of interactions at the global level and executed through message-passing at the local level. A word w in a global language represents then a possible run expected of a communicating system inducing an expected “local” behaviour on each participant A : the projection of w on A yields the sequence of *output* or *input* actions performed by A along the run w .

Our language-theoretic treatment is motivated mainly by the need for a general setting immune to syntactic restrictions. This naturally leads us to consider e.g., context-free choreographies (cf. Ex. 3.11). In fact, we strive for generality; basically *prefix-closure* is the only requirement we impose on FCL. The gist is that, if a sequence of interactions or of communications is an observable behaviour of a system, any prefix of the sequence should be observable as well. (We discuss some implications of relaxing prefix-closure in Section 8.) This allows us to consider partial executions as well as “complete” ones. We admit infinite words to account for diverging computations, ubiquitous in communication protocols.

Some g-languages cannot be faithfully executed by distributed components; consider $\{A \rightarrow B:m, A \rightarrow B:m \cdot C \rightarrow D:n\}$ that specifies a system where, if occurring, the interaction between C and D has to follow the one between A and B . Clearly, this is not possible if the participants act concurrently because C and D are not aware of when the interaction between A and B takes place.

Contributions & structure We summarise below our main contributions. (Proofs and further material can be found in [7].)

Section 2 introduces FCL (g-languages in Def. 2.1, l-languages in Def. 2.2) and adapts standard constructions from the literature. We consider synchronous interactions; the asynchronous case, albeit interesting, is scope for future work (cf. Section 8). In particular, we render communicating systems as sets of l-languages (Def. 2.3), while we borrow projections from choreographies and multiparty session types.

Section 3 considers correctness and completeness. An immediate consequence of our constructions is the completeness of systems projected from g-languages (Corollary 3.2). Correctness is more tricky; for it, Def. 3.3 introduces *closure under unknown information* (CUI). Intuitively, a g-language is CUI if it contains extensions of words with a single interaction whose participants cannot distinguish the extended word from other words of the language. Thm. 3.7 characterises correctness of projected systems in terms of CUI.

Section 4 shows how FCLs capture many relevant communication properties in a fairly uniform way.

Section 5 proposes *branch-awareness* (Def. 5.3) to ensure the communication properties defined in Section 4 (Thm. 5.6). Intuitively, branch-awareness requires each participant to “distinguish” words where its behaviour differs. Notably, we

separate the conditions for correctness from the ones for communication properties. Most approaches in the literature instead combine them into a single condition, which takes names such as well-branchedness or projectability [25]. Thus, these single conditions are stronger than each of CUI and branch-awareness.

Sections 6 and 7 illustrate the generality of FCLs on two case studies, respectively taken from multiparty session types [37] and choreography automata [6]. We remark that FCL can capture protocols that cannot be represented by regular g-languages such as the “task dispatching” protocol in Ex. 3.11. To the best of our knowledge this kind of protocols cannot be formalised in other approaches.

Section 8 draws some conclusions and discusses future work.

2 Formal Choreographic Languages

We briefly recall a few notions used through the paper. The sets of finite and infinite words on a given alphabet Σ are, respectively, denoted by Σ^* and Σ^ω , where an infinite word on Σ is a map from natural numbers to Σ (aka ω -word [38]). Let $\cdot\cdot$ be the concatenation operator on words and ε its neutral element. We write $a_0 \cdot a_1 \cdot a_2 \cdot \dots$ for the word mapping i to $a_i \in \Sigma$ for all natural numbers i . A language L on Σ is a subset of $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$. The prefix-closure of $L \subseteq \Sigma^\infty$ is $\text{pref}(L) = \{z \in \Sigma^\infty \mid \exists z' \in L : z \preceq z'\}$, where \preceq is the prefix relation; L is *prefix-closed* if $L = \text{pref}(L)$. A word z is *maximal* in a language $L \subseteq \Sigma^\infty$ if $z \preceq z'$ for $z' \in L$ implies $z' = z$. As usual we shall write $z \prec z'$ whenever $z \preceq z'$ and $z \neq z'$.

We shall deal with languages on particular alphabets, namely the alphabets of *interactions* Σ_{int} and of *actions* Σ_{act} whose definitions, borrowed from [6], are as follows⁴

$$\begin{aligned} \Sigma_{\text{int}} &= \{ \mathbf{A} \rightarrow \mathbf{B} : \mathbf{m} \mid \mathbf{A} \neq \mathbf{B} \in \mathfrak{P}, \mathbf{m} \in \mathfrak{M} \} && \text{ranged over by } \alpha, \beta, \dots \\ \Sigma_{\text{act}} &= \{ \mathbf{A} \mathbf{B} ! \mathbf{m}, \mathbf{A} \mathbf{B} ? \mathbf{m} \mid \mathbf{A} \neq \mathbf{B} \in \mathfrak{P}, \mathbf{m} \in \mathfrak{M} \} && \text{ranged over by } \mathbf{a}, \mathbf{b}, \dots \end{aligned}$$

where \mathfrak{P} is a fixed set of *participants* (or *roles*, ranged over by $\mathbf{A}, \mathbf{B}, \mathbf{X}$, etc.) and \mathfrak{M} is a fixed set of *messages* (ranged over by \mathbf{m}, \mathbf{x} , etc.); we take \mathfrak{P} and \mathfrak{M} disjoint. Let $\text{msg}(\mathbf{A} \rightarrow \mathbf{B} : \mathbf{m}) = \text{msg}(\mathbf{A} \mathbf{B} ! \mathbf{m}) = \text{msg}(\mathbf{A} \mathbf{B} ? \mathbf{m}) = \mathbf{m}$ and $\text{ptp}(\mathbf{A} \rightarrow \mathbf{B} : \mathbf{m}) = \text{ptp}(\mathbf{A} \mathbf{B} ! \mathbf{m}) = \text{ptp}(\mathbf{A} \mathbf{B} ? \mathbf{m}) = \{\mathbf{A}, \mathbf{B}\}$. These functions extend homomorphically to (sets of) words. The *subject* of $\mathbf{A} \mathbf{B} ! \mathbf{m}$ is the sender \mathbf{A} and the subject of $\mathbf{A} \mathbf{B} ? \mathbf{m}$ is the receiver \mathbf{B} . Words on $\Sigma_{\text{int}}^\infty$ (ranged over by w, w', \dots) are called *interaction words* while those on $\Sigma_{\text{act}}^\infty$ (ranged over by v, v', \dots) are called *words of actions*. Hereafter z, z', \dots range over $\Sigma_{\text{int}}^\infty \cup \Sigma_{\text{act}}^\infty$ and we use \mathcal{L} and \mathbb{L} to range over subsets of, respectively, $\Sigma_{\text{int}}^\infty$ and $\Sigma_{\text{act}}^\infty$.

A *global language* specifies the expected interactions of a system while a *local language* specifies the communication behaviour of participants.

Definition 2.1 (Global language). ^{def:charlang} A global language (g-language for short) is a prefix-closed language \mathcal{L} on $\Sigma_{\text{int}}^\infty$ such that $\text{ptp}(\mathcal{L})$ is finite.

⁴ These sets may be infinite; formal languages over infinite alphabets have been studied, e.g., in [4].

Definition 2.2 (Local language). ^{def:actLang} A local language (l-language for short) is a prefix-closed language \mathbb{L} on Σ_{act} such that $\mathit{ptp}(\mathbb{L})$ is finite. An l-language is A -local if its words have all actions with subject A .

As discussed in Section 1, l-languages give rise to *communicating systems*.

Definition 2.3 (Communicating system). ^{def:commSyst} Let $\mathcal{P} \subseteq \mathfrak{P}$ be a finite set of participants. A (communicating) system over \mathcal{P} is a map $S = (\mathbb{L}_A)_{A \in \mathcal{P}}$ assigning an A -local language $\mathbb{L}_A \neq \{\varepsilon\}$ such that $\mathit{ptp}(\mathbb{L}_A) \subseteq \mathcal{P}$ to each participant $A \in \mathcal{P}$.

By projecting a g-language \mathcal{L} on a participant A we obtain the A -local language describing the sequence of actions performed by A in the interactions involving A in the words of \mathcal{L} .

Definition 2.4 (Projection). ^{def:projection} The projection on A of an interaction $B \rightarrow C:m$ is computed by the function $\downarrow_A: \Sigma_{int} \times \mathfrak{P} \rightarrow \Sigma_{act} \cup \{\varepsilon\}$ defined by:

$$(A \rightarrow B:m) \downarrow_A = AB!m \quad (A \rightarrow B:m) \downarrow_B = AB?m \quad (A \rightarrow B:m) \downarrow_C = \varepsilon$$

and extended homomorphically to interaction words and g-languages. The projection of a g-language \mathcal{L} , written $\mathcal{L} \downarrow$, is the communicating system $(\mathcal{L} \downarrow_A)_{A \in \mathit{ptp}(\mathcal{L})}$.

Def. 2.4 recasts in our setting the notion of projection used, e.g., in [13,24].

Example 2.5. Let $\mathcal{L} = \mathit{pref}(\{C \rightarrow A:m \cdot A \rightarrow B:m, C \rightarrow B:m \cdot A \rightarrow B:m, C \rightarrow A:m \cdot C \rightarrow B:m\})$. ^{ex:simple} By Def. 2.4, we have $\mathcal{L} \downarrow = (\mathcal{L} \downarrow_x)_{x \in \{A,B,C\}}$ where $\mathcal{L} \downarrow_A = \{\varepsilon, CA?m, CA?m \cdot AB!m, AB!m\}$, $\mathcal{L} \downarrow_B = \{\varepsilon, AB?m, CB?m, CB?m \cdot AB?m\}$, and $\mathcal{L} \downarrow_C = \{\varepsilon, CA!m, CB!m, CA!m \cdot CB!m\}$. \diamond

We consider a *synchronous* semantics of communicating systems, similarly to other choreographic approaches such as [12,13,15,37]. Intuitively, a choreographic word is in the semantics iff its projection on each participant A yields a word in the local language of A .

Definition 2.6 (Semantics). ^{def:syncSem} Given a system S over \mathcal{P} , the set

$$\llbracket S \rrbracket = \{w \in \Sigma_{int}^\infty \mid \mathit{ptp}(w) \subseteq \mathcal{P} \wedge \forall A \in \mathcal{P} : w \downarrow_A \in S(A)\}$$

is the (synchronous) semantics of S .

Notice that the above definition coincides with the *join* operation in [18], used in realisability conditions for an asynchronous setting.

Example 2.7. The semantics $\llbracket \mathcal{L} \downarrow \rrbracket$ of the system $\mathcal{L} \downarrow$ in Ex. 2.5 is the prefix closure of $\{C \rightarrow A:m \cdot A \rightarrow B:m, C \rightarrow B:m \cdot A \rightarrow B:m, C \rightarrow A:m \cdot C \rightarrow B:m \cdot A \rightarrow B:m\}$. ^{ex:simple2} \diamond

Two interactions α and β are *independent* (in symbols $\alpha \parallel \beta$) when $\mathit{ptp}(\alpha) \cap \mathit{ptp}(\beta) = \emptyset$. Informally, independent interactions can be swapped. The concurrency closure on infinite words is delicate. One in fact has to allow infinitely many swaps while avoiding that they make an interaction disappear by pushing it infinitely far away. Technically, we consider Mazurkiewicz's traces [33] on Σ_{int} with independence relation $\alpha \parallel \beta$:

Definition 2.8 (Concurrency closure). ^{def:cclos} Let \sim be the reflexive and transitive closure of the relation \equiv on finite interaction words defined by $w \alpha \beta w' \equiv w \beta \alpha w'$ where $\alpha \parallel \beta$. Following [19, Def. 2.1], \sim extends to Σ_{int}^ω by defining

$$\text{for all } w, w' \in \Sigma_{int}^\omega : \quad w \sim w' \iff w \ll w' \text{ and } w' \ll w$$

where $w \ll w'$ iff for each finite prefix w_1 of w there are a finite prefix w'_1 of w' and a g -word $\hat{w} \in \Sigma_{int}^*$ such that $w_1 \cdot \hat{w} \sim w'_1$. A g -language \mathcal{L} is concurrency closed (*c-closed for short*) if it coincides with its concurrency closure, namely $\mathcal{L} = \{w \in \Sigma_{int}^\omega \mid \exists w' \in \mathcal{L} : w \sim w'\}$.

Semantics of systems are naturally c-closed since in a distributed setting independent events can occur in any order. Indeed

thmt@ccsem@data

thmp@pcpan

Proposition 2.9. *Let S be a system. Then $\llbracket S \rrbracket$ is c-closed.*

The intuition that g -languages, equipped with the projection and semantic functions of Def. 2.4 and Def. 2.6, do correspond to a natural syntax and semantics for the abstract notion of choreography, can be strengthened by showing that these functions form a Galois connection.

Let us define $\mathbb{G} = \{\mathcal{L} \mid \mathcal{L} \text{ is a } g\text{-language}\}$ and $\mathbb{S} = \{S \mid S \text{ is a system}\}$. Moreover, given $S, S' \in \mathbb{S}$, we define $S \subseteq S'$ if $S(\mathbf{A}) \subseteq S'(\mathbf{A})$ for each \mathbf{A} .

thmt@gc@data

Proposition 2.10. *The functions \downarrow and $\llbracket _ \rrbracket$ form a (monotone) Galois connection between the posets (\mathbb{G}, \subseteq) and (\mathbb{S}, \subseteq) , namely, \downarrow and $\llbracket _ \rrbracket$ are monotone functions such that, given $\mathcal{L} \in \mathbb{G}$ and $S \in \mathbb{S}$:*

thmp@gc

$$\mathcal{L} \downarrow \subseteq S \iff \mathcal{L} \subseteq \llbracket S \rrbracket$$

Notice that, by Prop. 2.10, $\mathcal{L} \downarrow \subseteq S$ can be understood as “ \mathcal{L} can be realized by S ” according to the notion of realisability frequently used in the literature, namely that all behaviours of the choreography are possible for the system.

It is well-known that, given a Galois connection (f_*, f^*) the function $\mathbf{cl} = f^* \circ f_*$ is a closure operator namely, it is monotone ($x \leq y \implies \mathbf{cl}(x) \leq \mathbf{cl}(y)$), extensive ($x \leq \mathbf{cl}(x)$), and idempotent ($\mathbf{cl}(x) = \mathbf{cl}(\mathbf{cl}(x))$). In our setting $\mathbf{cl}(_) = \llbracket _ \downarrow \rrbracket$, hence the above boils down to the following corollary:

fact:clgc

Corollary 2.11. *For all g -languages $\mathcal{L}, \mathcal{L}' \in \mathbb{G}$,*

monotonicity: $\mathcal{L} \subseteq \mathcal{L}' \implies \llbracket \mathcal{L} \downarrow \rrbracket \subseteq \llbracket \mathcal{L}' \downarrow \rrbracket$,

extensiveness: $\mathcal{L} \subseteq \llbracket \mathcal{L} \downarrow \rrbracket$,

idempotency: $\llbracket \mathcal{L} \downarrow \rrbracket = \llbracket \llbracket \mathcal{L} \downarrow \rrbracket \downarrow \rrbracket$.

As we shall see, extensiveness coincides with completeness (Def. 3.1) and, together with monotonicity, implies *harmonicity* (Def. 4.1).

3 Correctness and Completeness

A g-language specifies the expected communication behaviour of a system made of several components. We now define properties relating a communicating system (i.e., a set of l-languages) with a specification (i.e., a g-language). axp@eisi

Definition 3.1 (Correctness and completeness). ^{def:cc} A system S is correct (resp. complete) w.r.t. a g-language \mathcal{L} if $\llbracket S \rrbracket \subseteq \mathcal{L}$ (resp. $\llbracket S \rrbracket \supseteq \mathcal{L}$). tmt@completeness@data

Correctness and completeness are related to existing notions. For instance, in the literature on multiparty session types (see, e.g., the survey [25]) correctness is analogous to *subject reduction* and completeness to *session fidelity*. Notice that by Prop. 2.10, we can interpret $\mathcal{L} \downarrow \subseteq S$ as a characterisation for completeness of S w.r.t. \mathcal{L} .

We discuss now how to ensure correctness and completeness “by construction”. Completeness is trivial: it holds for any projected system and coincides with the extensiveness property of the closure operator associated to the Galois connection defined in Section 2. tmt@completeness

Corollary 3.2. *The projection of a g-language \mathcal{L} is complete w.r.t. \mathcal{L} .* tmt@completeness

We show now how correctness can be characterised as a closure property.

Definition 3.3 (CUI). ^{def:closedness} A g-language \mathcal{L} is closed under unknown information (in symbols $\text{cui}(\mathcal{L})$) if, for all finite words $w_1 \cdot \alpha, w_2 \cdot \alpha \in \mathcal{L}$ with the same final interaction $\alpha = A \rightarrow B : m \in \Sigma_{\text{int}}$, $w \cdot \alpha \in \mathcal{L}$ for all $w \in \mathcal{L}$ such that $w \downarrow_A = w_1 \downarrow_A$ and $w \downarrow_B = w_2 \downarrow_B$. ex:notCUI

Intuitively, participants cannot distinguish words with the same projection on their role. Hence, if two participants A and B find words w_1 and w_2 compatible with another word w , and interaction $A \rightarrow B : m$ can occur after both w_1 and w_2 , then it should be enabled also after w . Indeed, A (resp. B) cannot know whether the current word is w or w_1 (resp. w_2), hence A and B are willing to take $A \rightarrow B : m$, which can thus happen at the system level. Closure under unknown information (CUI for short) lifts this requirement at the level of g-language.

Example 3.4. The language \mathcal{L} in Ex. 2.5 is not CUI because it contains the words ex:notCUI

$$w_1 \cdot \alpha = C \rightarrow A : m \cdot A \rightarrow B : m \quad w_2 \cdot \alpha = C \rightarrow B : m \cdot A \rightarrow B : m \quad \text{and} \quad w = C \rightarrow A : m \cdot C \rightarrow B : m$$

and A cannot distinguish between w_1 and w while B cannot distinguish between w_2 and w ; nonetheless $w \cdot A \rightarrow B : m = C \rightarrow A : m \cdot C \rightarrow B : m \cdot A \rightarrow B : m \notin \mathcal{L}$. Notice that $w \cdot A \rightarrow B : m \in \llbracket \mathcal{L} \rrbracket$, hence $\mathcal{L} \not\supseteq \llbracket \mathcal{L} \rrbracket$. ◇

The language in Ex. 3.4 is not the semantics of any system, in fact languages obtained as semantics of a communicating system are always CUI. tmt@semiscui@data

Proposition 3.5 (Semantics is CUI). ^{thm@semiscui} For all systems S , $\llbracket S \rrbracket$ is CUI.

The next property connects finite and infinite words in a language; it corresponds to the closure under the limit operation used in ω -languages [17,38].

Definition 3.6 (Continuity). ^{def, continuity} A language L on an alphabet Σ is continuous if $z \in L$ for all $z \in \Sigma^\omega$ such that $\text{pref}(z) \cap L$ is infinite.

This notion of continuity, besides being quite natural, is the most suitable for our purposes among the possible ones [36]. Intuitively, a language L is continuous if an ω -word is in L when infinitely many of its approximants (i.e., finite prefixes) are in L . A g-language \mathcal{L} is *standard or continuous* (sc-language, for short) if either $\mathcal{L} \subseteq \Sigma_{\text{int}}^*$ or \mathcal{L} is continuous. Notice that for prefix-closed languages for all $z \in L^\omega$ we have that $\text{pref}(z) \cap L$ is infinite iff $\text{pref}(z) \subseteq L$.

Closure under unknown information characterises correct projected systems. ^{thmt@correctness@data}

Theorem 3.7 (Characterisation of correctness). ^{this@correctness} If $\mathcal{L} \downarrow$ is correct w.r.t. \mathcal{L} then $\text{cui}(\mathcal{L})$ holds. If \mathcal{L} is an sc-language and $\text{cui}(\mathcal{L})$ then $\mathcal{L} \downarrow$ is correct w.r.t. \mathcal{L} .

Notice that CUI is defined in terms of g-languages only, hence checking CUI does not require to build the corresponding system. Also, strengthening the precondition of Def. 3.3 with the additional requirement $w_1 = w_2$ would invalidate Thm. 3.7. Indeed, the language in Ex. 2.5 would become CUI but not correct. The next example shows that the continuity condition in Thm. 3.7 is necessary for languages containing infinite g-words.

Example 3.8 (Continuity matters). The CUI language

$$\mathcal{L} = \text{pref}\left(\bigcup_{i \geq 0} \{A \rightarrow B: l \cdot B \rightarrow C: n \cdot (C \rightarrow D: n)^i\} \cup \{A \rightarrow B: r \cdot B \rightarrow C: n \cdot (C \rightarrow D: n)^\omega\}\right)$$

does contain an infinite word but it is not continuous. The projection of \mathcal{L} is not correct because its semantics contains the g-word $A \rightarrow B: l \cdot B \rightarrow C: n \cdot (C \rightarrow D: n)^\omega \notin \mathcal{L}$ since the projections of C and D can exchange infinitely many messages n due to the infinite g-word of \mathcal{L} regardless whether A and B exchange l or r . \diamond

Notice that, since $\mathcal{L} \subseteq \llbracket \mathcal{L} \downarrow \rrbracket$ always holds, Thm. 3.7 implies that $\text{cui}(\mathcal{L})$ characterises the languages \mathcal{L} such that $\mathcal{L} = \llbracket \mathcal{L} \downarrow \rrbracket$. Besides, the following corollary descends from Thm. 3.7. ^{thmt@smallest@data}

Corollary 3.9. ^{thmt@smallest} For each sc-language \mathcal{L} , $\text{cl}(\mathcal{L})$ is the smallest CUI sc-language containing \mathcal{L} .

CUI ensures that continuous g-languages are c-closed. ^{thmt@continuousiscc@data}

Proposition 3.10. ^{thmt@continuousiscc} If \mathcal{L} is an sc-language and $\text{cui}(\mathcal{L})$, then \mathcal{L} is c-closed.

Hence, an sc-language cannot be CUI unless it is c-closed.

As recalled before, in many choreographic formalisms (such as [5,25,14,9,18]) the correctness and completeness of a projected system, namely $\mathcal{L} = \llbracket \mathcal{L} \downarrow \rrbracket$ (together with some forms of liveness and deadlock-freedom properties), is guaranteed by *well-branchedness* conditions. Most of such conditions guarantee, informally, that participants reach consensus on which branch to take when choices arise. For instance, a well-branchedness condition could be that, at each choice,

there is a unique participant deciding the branch to follow during a computation and that such participant informs each other participant. Such a condition is actually not needed to prove $\mathcal{L} = \llbracket \mathcal{L} \downarrow \rrbracket$. In fact the g-language obtained by adding the word w of Ex. 3.4 to the language of Ex. 2.5 is CUI, without being well-branched in the above sense. Indeed, after the interaction $C \rightarrow A:m$, there is a branching in the projected system, since both the interactions $C \rightarrow B:m$ and $A \rightarrow B:m$ can be performed. However, these interactions do not have the same sender.

The next example exhibits a non-regular CUI g-language of finite words. By Thm. 3.7 and Corollary 3.2, the projected system is correct and complete.

Example 3.11 (Task dispatching). ^{ex:parenthesis} As soon as a server (**S**) communicates its availability (**a**), a dispatcher (**D**) sends a task (**t**) to **S**. The server either processes the task directly and sends back the resulting data (**d**) to **D** or sends the task to participant **H** for some pre-processing, aiming at resuming it later on. Indeed, after communicating a result to **D**, the server can resume (**r**) a previous task (if any) from **H**, process it, and send the result to **D**. The server eventually stops by sending **s** to both **D** and **H**; this can happen only when all dispatched tasks have been processed.

This protocol corresponds to the g-language $\mathcal{L} = \text{pref}(\mathfrak{L})$, where \mathfrak{L} is the (non-regular) language generated by the following context-free grammar.

$$S ::= S' \cdot S \rightarrow D:s \cdot S \rightarrow H:s \quad S' ::= S \rightarrow D:a \cdot D \rightarrow S:t \cdot S \rightarrow H:t \cdot S' \cdot S \rightarrow H:r \cdot H \rightarrow S:r \cdot S \rightarrow D:d \cdot S' \\ \mid S \rightarrow D:a \cdot D \rightarrow S:t \cdot S \rightarrow D:d \cdot S' \quad \mid \varepsilon$$

Since **S** is involved in all the interactions of \mathcal{L} , for each pair of words $w, w' \in \mathcal{L}$: $w \downarrow_S = w' \downarrow_S$ iff $w = w'$. Now, if $w_1 \alpha, w_2 \alpha, w \in \mathcal{L}$ satisfy the required conditions for CUI then either $w_1 \downarrow_S = w \downarrow_S$ or $w_2 \downarrow_S = w \downarrow_S$, since $S \in \text{ptp}(\alpha)$. Hence $\text{cui}(\mathcal{L})$ trivially holds. \diamond

The language in Ex. 3.11 is non-regular since it has the same structure of a language of well-balanced parenthesis. Remarkably, this implies that the g-language cannot be expressed in any other choreographic model we are aware of. The argument used to show $\text{cui}(\mathcal{L})$ in Ex. 3.11 proves the following.

Proposition 3.12. *If there exists a participant involved in all the interactions of a g-language \mathcal{L} then $\text{cui}(\mathcal{L})$ holds.*

4 Communication Properties

Besides correctness and completeness, other properties could be of interest. For instance, one would like to ensure that participants eventually interact, if they are willing to. We consider a few properties, informally described as follows.

Harmonicity (HA): each sequence of communications that a participant is able to perform can be executed in some computation of the system.

Lock-freedom (LF): if a participant has pending communications to make on an ongoing computation, then there is a continuation of the computation involving that participant.

Strong lock-freedom (SLF): if a participant has pending communications to make on an ongoing computation, then each maximal continuation of the computation involves that participant.

Starvation-freedom (SF): if a participant has pending communications to make on an ongoing computation, then each infinite continuation of the computation involves that participant.

Deadlock-freedom (DF): in all completed computations each participant has no pending actions.

We now formalise the properties above.

Definition 4.1 (Communication properties). ^{def:properties} Let S be a system on \mathcal{P} .

HA S is harmonic if $S(\mathbf{A}) \subseteq \llbracket S \rrbracket \downarrow_{\mathbf{A}}$ for each $\mathbf{A} \in \mathcal{P}$. ^{def:llive}

LF S is lock free if, for each finite word $w \in \llbracket S \rrbracket$ and participant $\mathbf{A} \in \mathcal{P}$, if $w \downarrow_{\mathbf{A}}$ is not maximal in $S(\mathbf{A})$ then there is a word w' such that $ww' \in \llbracket S \rrbracket$ and $w' \downarrow_{\mathbf{A}} \neq \varepsilon$. ^{def:wlive}

SLF S is strongly lock free if, for each finite $w \in \llbracket S \rrbracket$ and participant $\mathbf{A} \in \mathcal{P}$, if $w \downarrow_{\mathbf{A}}$ is not maximal in $S(\mathbf{A})$ then for each word w' such that ww' is maximal in $\llbracket S \rrbracket$ we have $w' \downarrow_{\mathbf{A}} \neq \varepsilon$. ^{def:sfl}

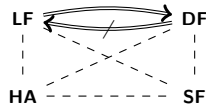
SF S is starvation free if, for each finite $w \in \llbracket S \rrbracket$ and participant $\mathbf{A} \in \mathcal{P}$, if $w \downarrow_{\mathbf{A}}$ is not maximal in $S(\mathbf{A})$ then $w' \downarrow_{\mathbf{A}} \neq \varepsilon$ for each infinite word w' such that $ww' \in \llbracket S \rrbracket$. ^{def:lf}

DF S is deadlock free if, for each finite and maximal word $w \in \llbracket S \rrbracket$ and participant $\mathbf{A} \in \mathcal{P}$, $w \downarrow_{\mathbf{A}}$ is maximal in $S(\mathbf{A})$. ^{def:deadlockfree}

Barred for harmonicity, these properties appear in the literature under different names in various contexts. For instance, the notion of lock-freedom in [5] corresponds to ours, which in turn corresponds to the notion of liveness in [32,29] in a channel-based synchronous communication setting. Likewise, the notion of strong lock-freedom in [37] corresponds to ours and, under fair scheduling, to the notion of lock-freedom in [28]. As a final example, the definition of deadlock-freedom in its (equivalent) contrapositive form, coincides with the notion of progress as defined for synchronous processes in [35,23]. Harmonicity, introduced in the present paper, assures that no behaviour of a participant can be taken out from a system without affecting the overall behaviour of the system itself. Notice that the inverse of harmonicity, $\llbracket S \rrbracket \downarrow_{\mathbf{A}} \subseteq S(\mathbf{A})$, holds by construction.

The next proposition highlights the relations among our properties. ^{thmt@systemproperties@data}

Proposition 4.2. ^{thmt@systemproperties} The following relations hold among the properties in Def. 4.1



where implication does not hold in any direction between properties connected by dashed lines

Moreover, $DF \wedge SF \Leftrightarrow SLF$.

5 Communication Properties by Construction

Harmonicity is the only property in Def. 4.1 guaranteed by projection on any system. This can be obtained as a simple consequence of Corollary 3.2. ^{sec:projprop} ^{thmt@wrlive@data}

Corollary 5.1. *If \mathcal{L} is a g-language then $\mathcal{L}\downarrow$ is harmonic.* thhh@ufliweb

The other properties require some conditions on systems to be enjoyed by $\mathcal{L}\downarrow$. Basically, we will strengthen CUI which is too weak. For instance, $\text{cui}(\mathcal{L})$ does imply neither deadlock-freedom nor lock-freedom for $\mathcal{L}\downarrow$.

Example 5.2 (CUI $\not\Rightarrow$ DF). ^{ex-cloesodl} Consider the following words

$$w = A \rightarrow C.l \cdot A \rightarrow B.m \cdot A \rightarrow C.m \quad \text{and} \quad w' = A \rightarrow C.r \cdot A \rightarrow B.m \cdot B \rightarrow C.m$$

It is easy to check that the g-language $\mathcal{L} = \text{pref}(\{w, w'\})$ is CUI. Informally, $\text{cui}(\mathcal{L})$ holds because C can ascertain which of its last actions to execute from the first input. So, Corollary 3.2 and Thm. 3.7 ensure that $\mathcal{L} = \llbracket \mathcal{L}\downarrow \rrbracket$. However, $\mathcal{L}\downarrow$ is not deadlock-free. In particular, $w \in \mathcal{L} = \llbracket \mathcal{L}\downarrow \rrbracket$ is a deadlock since it is a finite maximal word whose projection on B , namely $w \downarrow_B = AB^?m$, is not maximal in $\mathcal{L}\downarrow_B$ because $w' \downarrow_B = AB^?m \cdot BC!m \in \mathcal{L}\downarrow_B$.

By Prop. 4.2, the system above is also non lock-free. \diamond

In many models (cf. [25]) in order to ensure, besides other properties, also the correctness of $\mathcal{L}\downarrow$, a condition called *well-branchedness* is required. We identify a notion weaker than well-branchedness, which by analogy we dub *branch-awareness* (BA for short).

Definition 5.3 (Branch-awareness). ^{def:ba} A participant X distinguishes two g-words $w_1, w_2 \in \Sigma_{int}^\infty$ if

$$w_1 \downarrow_X \neq w_2 \downarrow_X \quad \text{and} \quad w_1 \downarrow_X \not\prec w_2 \downarrow_X \quad \text{and} \quad w_2 \downarrow_X \not\prec w_1 \downarrow_X .$$

A g-language \mathcal{L} on \mathcal{P} is branch-aware if each $X \in \mathcal{P}$ distinguishes all maximal words in \mathcal{L} whose projections on X differ.

Example 5.4. The language $\mathcal{L} = \text{pref}(\{w, w'\})$ with $w = A \rightarrow C.l \cdot A \rightarrow B.m \cdot A \rightarrow C.m$ and $w' = A \rightarrow C.r \cdot A \rightarrow B.m \cdot B \rightarrow C.m$ from Ex. 5.2 is not branch-aware, since $w \downarrow_B = AB^?m$ and $w' \downarrow_B = AB^?m \cdot BC!m$, hence $w \downarrow_B \neq w' \downarrow_B$ but $w \downarrow_B \prec w' \downarrow_B$. \diamond

Condition $w_1 \downarrow_X \neq w_2 \downarrow_X$ in Def. 5.3 is not strictly needed to define BA, but it makes the notion of ‘distinguishes’ more intuitive. Equivalently, as shown in Prop. 5.5 below, a participant X distinguishes two branches if, after a common prefix, X is actively involved in both branches, performing different interactions. thmt@@distinguish@data

Proposition 5.5. ^{thmt@@distinguish} Participant X distinguishes two g-words $w_1, w_2 \in \Sigma_{int}^\infty$ iff ^{thmt@@baconseq@data} there are $w'_1 \cdot \alpha_1 \preceq w_1$ and $w'_2 \cdot \alpha_2 \preceq w_2$ such that $w'_1 \downarrow_X = w'_2 \downarrow_X$ and $\alpha_1 \downarrow_X \neq \alpha_2 \downarrow_X$.

The notions of well-branchedness in the literature [25] additionally impose that $\alpha_1 \downarrow_X$ and $\alpha_2 \downarrow_X$ in the above proposition are input actions, but for a (unique) participant (a.k.a., the *selector*) which is required to have different outputs.

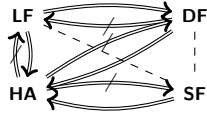
In our case, BA is not needed for correctness, but it is nevertheless useful to prove the communication properties presented in Section 4. \diamond

Theorem 5.6 (Consequences of BA). ^{thmt00hansggcg} Let \mathcal{L} be a branch-aware and CUI sc-language. Then $\mathcal{L}\downarrow$ satisfies all the properties in Def. 4.1.

Example 5.7 (The task dispatching protocol is branch aware). ^{ex:parenthesis2} In order to show that the g-language \mathcal{L} in Ex. 3.11 is branch-aware, we first notice that each maximal word in \mathcal{L} ends with the interactions $S \rightarrow D:s \rightarrow S \rightarrow H:s$. If \mathcal{L} were not branch-aware, there should be two maximal words $w \cdot S \rightarrow D:s \rightarrow S \rightarrow H:s$ and $w' \cdot S \rightarrow D:s \rightarrow S \rightarrow H:s$ and a participant $X \in \text{ptp}(\mathcal{L})$ such that $(w \cdot S \rightarrow D:s \rightarrow S \rightarrow H:s)\downarrow_X < (w' \cdot S \rightarrow D:s \rightarrow S \rightarrow H:s)\downarrow_X$. This is impossible, since w and w' are both generated by the non terminal symbol S' and hence cannot contain the message s . \diamond

Prop. 4.2 refines as follows when restricting to projections of g-languages.

Proposition 5.8. ^{thmt00scmp@pcc@br@data} When considering only systems which are projections of g-languages the following relations hold among the properties in Def. 4.1



where implication does not hold in any direction between properties connected by dashed lines

Moreover, $DF \wedge SF \Leftrightarrow SLF$.

It is not difficult to show that branch-awareness actually characterises SLF for systems obtained by projecting CUI languages. ^{thmt00bachar@data}

Proposition 5.9 (Branch-awareness characterises SLF). ^{thmt00bachar} A CUI g-language \mathcal{L} is branch-aware iff $\mathcal{L}\downarrow$ is strongly lock-free.

6 Global Types as Choreographic Languages

The global types of [37] are our first case study. We recall global types adapting some of the notation in [37] to our setting. Informally, a global type $A \rightarrow B : \{m_i.G_i\}_{1 \leq i \leq n}$ specifies a protocol where participant A must send to B a message m_i for some $1 \leq i \leq n$ and then, depending on which m_i was chosen by A , the protocol continues as G_i . Global types and multiparty sessions are defined in [37] in terms of the following grammars:

$$\begin{array}{lll} G ::=^{\text{co}} \text{end} & P ::=^{\text{co}} \mathbf{0} & \mathcal{M} ::= A \triangleright P \\ | A \rightarrow B : \{m_i.G_i\}_{1 \leq i \leq n} & | A?\{m_i.P_i\}_{1 \leq i \leq n} & | \mathcal{M} \mid \mathcal{M} \\ & | A!\{m_i.P_i\}_{1 \leq i \leq n} & \end{array}$$

respectively for *pre-global types*, *pre-processes*, and *pre-multiparty sessions*. The first two grammars are interpreted coinductively, that is their solutions are both minimal and maximal fixpoints (the latter corresponding to infinite trees) and all messages m_i are pairwise different. A pre-global type G (resp. pre-process P) is a *global type* (resp. *process*) if its tree representation is *regular*, namely it has finitely many distinct sub-trees. A *multiparty session* (MPS for short) is a pre-multiparty session such that (a) in $A \triangleright P$, participant A does not occur in process P and (b) in $A_1 \triangleright P_1 \mid \dots \mid A_n \triangleright P_n$, participants A_i are pairwise different.

The semantics of global types is the LTS induced by

$$A \rightarrow B : \{m_i.G_i\}_{1 \leq i \leq n} \xrightarrow{A \rightarrow B.m_i} G_i \quad R \rightarrow S : \{m_i.G_i\}_{1 \leq i \leq n} \xrightarrow{A \rightarrow B.m} R \rightarrow S : \{m_i.G'_i\}_{1 \leq i \leq n}$$

where in the latter rule $\{A, B\} \cap \{R, S\} = \emptyset$ and for each $1 \leq i \leq n$, $G_i \xrightarrow{A \rightarrow B.m} G'_i$. A *branch* is a set $\{m_i.P_i\}_{1 \leq i \leq n}$ where messages m_i are pairwise distinct.

The semantics for MPSs is the LTS defined by the following rule

$$A \triangleright B!(\{m.P\} \uplus A) \mid B \triangleright A?(\{m.P'\} \uplus A') \mid \mathcal{M} \xrightarrow{A \rightarrow B.m} A \triangleright P \mid B \triangleright P' \mid \mathcal{M} \quad (1) \text{eq:mpsem}$$

where \uplus is the union of branches defined only on branches with disjoint sets of messages. Rule (1) applies only if the messages in A' include those in A , which is the case for MPSs obtained by projection, defined below.

Definition 6.1 (Projection [37, Definition 3.4]). *The projection of G on a participant X such that the depths of its occurrences in G are bounded is the partial function $G \downarrow_X$ coinductively defined by $\text{end} \downarrow_X = \mathbf{0}$ and, for a global type $G = A \rightarrow B : \{m_i.G_i\}_{1 \leq i \leq n}$, by:*

$$G \downarrow_X = \begin{cases} \mathbf{0} & \text{if } X \text{ is not a participant of } G \\ B!\{m_i.G_i \downarrow_X\}_{1 \leq i \leq n} & \text{if } X = A \\ A?\{m_i.G_i \downarrow_X\}_{1 \leq i \leq n} & \text{if } X = B \\ G_1 \downarrow_X & \text{if } X \notin \{A, B\} \text{ and } n = 1 \\ S?(A_1 \uplus \dots \uplus A_n) & \text{if } X \notin \{A, B\}, n > 1, \text{ and } \forall 1 \leq i \leq n : G_i \downarrow_X = S?A_i \end{cases}$$

The global type G is projectable⁵ if $G \downarrow_X$ is defined for all participants X of G , in which case $G \downarrow$ denotes the corresponding MPS.

The g-language $\mathcal{L}(G)$ associated to a global type G is the concurrency and prefix closure of $\mathcal{L}'(G)$, that is $\mathcal{L}(G) = \text{pref}(\{w \in \Sigma_{\text{int}}^\infty \mid \exists w' \in \mathcal{L}'(G) : w \sim w'\})$ where $\mathcal{L}'(G)$ is coinductively defined as follows:

$$\mathcal{L}'(\text{end}) = \{\varepsilon\} \quad \text{and} \quad \mathcal{L}'(A \rightarrow B : \{m_i.G_i\}_{1 \leq i \leq n}) = \bigcup_{1 \leq i \leq n} \{A \rightarrow B.m_i.w \mid w \in \mathcal{L}'(G_i)\}$$

We define the l-language $\mathbb{L}(B \triangleright P)$ associated to a named process $B \triangleright P$ as the prefix closure of $\mathbb{L}'(B \triangleright P)$ which, letting $\star \in \{?, !\}$, is defined by

$$\mathbb{L}'(B \triangleright \mathbf{0}) = \{\varepsilon\} \quad \text{and} \quad \mathbb{L}'(B \triangleright A \star \{m_i.P_i\}_{1 \leq i \leq n}) = \bigcup_{1 \leq i \leq n} \{AB \star m_i.w \mid w \in \mathbb{L}'(P_i)\}$$

The system associated to an MPS is defined as the following map:

$$S(A_1 \triangleright P_1 \mid \dots \mid A_n \triangleright P_n) = \{A_i \mapsto \mathbb{L}(A_i \triangleright P_i) \mid 1 \leq i \leq n\}$$

Our constructions capture relevant properties of the global types in [37]. First, we relate projectability (cf. Def. 6.1) and our properties. ^{thmt@wfb@data}

Proposition 6.2. *If G is a projectable global type then $\mathcal{L}(G)$ is a CUI and branch-aware sc-language.* ^{phmp@ufba}

This yields the following correspondences between the two frameworks. ^{thmt@severi@data}

Proposition 6.3. *Given a projectable global type G ,* ^{thmt@reperi}

⁵ In [37], projectability embeds well-branchedness.

$$\mathcal{L}(G) = \{ w \mid G \xrightarrow{w} \} \quad (2)^{eq:langds} \quad \llbracket S(G \downarrow) \rrbracket = \{ w \mid G \downarrow \xrightarrow{w} \} \quad (3)^{eq:projds}$$

Projectable global types are proved strongly lock-free in [37]. The following result corresponds to [37, Theorem 4.7].

Corollary 6.4. *$S(G \downarrow)$ is strongly lock-free for any projectable G .* cor:WFLtoLF

The symmetry between senders and receivers in CUI and branch-awareness allows for an immediate generalisation of the projection in Def. 6.1 by extending the last case with the clause:

$$S!(A_1 \uplus \dots \uplus A_n) \quad \text{if } X \notin \{A, B\}, n > 1, \text{ and } \forall 1 \leq i \leq n : G_i \downarrow_X = S!A_i$$

Corollary 6.4 still holds for this generalised definition of projection.

7 Choreography Automata

Recently we introduced *choreography automata* (c-automata) [6] as an expressive and flexible model of global specifications. A c-automaton $\mathbb{C}A = \langle \mathcal{S}, q_0, \Sigma_{\text{int}}, \rightarrow \rangle$ is a finite-state automaton whose transition relation is labelled in Σ_{int} , namely $\rightarrow \subseteq \mathcal{S} \times \Sigma_{\text{int}} \times \mathcal{S}$ (cf. [7, Def. 8.2]: for the sake of space most of the technical details of this section are in [7]). Observe that the set \mathcal{P} of participants of $\mathbb{C}A$ is necessarily finite. We have some immediate connection between c-automata and FCL by taking as the language $\mathcal{L}(\mathbb{C}A)$ of $\mathbb{C}A$ the set of words obtained by concatenating the labels on any of its paths (including infinite paths, cf. [7, Def. 8.3]). In fact $\mathcal{L}(\mathbb{C}A)$ is a continuous g-language, that is it is prefix-closed (cf. [7, Prop. 8.4]). sec:chor-automata

The local behaviour of a participant $A \in \mathcal{P}$ can be straightforwardly obtained by projecting c-automata on *communicating finite-state machines* (CFSMs) [11]. Basically, a CFSM is a finite-state automaton whose transitions are labelled in Σ_{act} (cf. [7, Def. 8.1]). Formally, the *projection* of a c-automaton $\mathbb{C}A$ on A , written $\mathbb{C}A \downarrow_A$, is obtained by determinising up-to-language equivalence the *intermediate* automaton

$$A_A = \langle \mathcal{S}, q_0, \Sigma_{\text{act}} \cup \{ \varepsilon \}, \{ q \xrightarrow{\lambda_A} q' \mid q \xrightarrow{\lambda} q' \} \rangle$$

Finally, $\mathbb{C}A \downarrow = (\mathbb{C}A \downarrow_A)_{A \in \mathcal{P}}$ is the *projection of $\mathbb{C}A$* (cf. [7, Def. 8.5]).

By applying the definition of language of c-automaton to CFSMs we can associate an l-language $\mathbb{L}(M)$ to each CFSM M (cf. [7, Def. 8.3]). Projections of c-automata and of the corresponding g-languages are related: $\mathbb{L}(\mathbb{C}A \downarrow_A) = \mathcal{L}(\mathbb{C}A) \downarrow_A$ (cf. [7, Prop. 8.8]).

The synchronous behaviour of a system of CFSMs $(M_A)_{A \in \mathcal{P}}$ can be given as an LTS where states are maps assigning a state in M_A to each $A \in \mathcal{P}$ and transitions are labelled by interactions (or by ε). Intuitively, given a configuration s , if M_A and M_B have respectively transitions $s(A) \xrightarrow{AB!m} q'_A$ and $s(B) \xrightarrow{AB?m} q'_B$ then $s \xrightarrow{A \rightarrow B:m} s[A \mapsto q'_A, B \mapsto q'_B]$, where $f[x \mapsto y]$ denotes the update of f on x

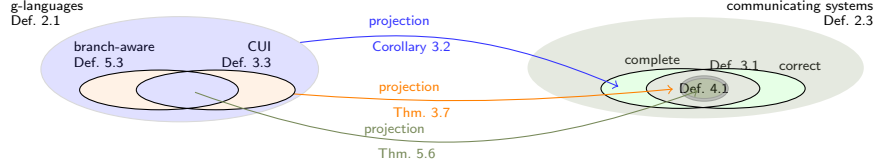


Fig. 1. Contributions of the paper

fig:contributions

with y . Likewise, $s(\mathbf{A}) \xrightarrow{\varepsilon} q'_A$ in M_A implies $s \xrightarrow{\varepsilon} s[\mathbf{A} \mapsto q'_A]$. Observing that $\mathbb{C}\mathbf{A}\downarrow$ is ε -free, the LTS of $\mathbb{C}\mathbf{A}\downarrow$ is a c -automaton and its language coincides with the g -language of the system $\{\mathbb{L}(\mathbb{C}\mathbf{A}\downarrow_x)\}_{x \in \mathcal{P}}$ (cf. [7, Prop. 8.7]).

The communication properties of a system of CFSMs S on \mathcal{P} considered in [6] are liveness, lock-freedom, and deadlock-freedom. We give an intuition of such properties (see [7, Def. 8.9] for a precise account).

- S is *live* when each reachable configuration where a participant $\mathbf{A} \in \mathcal{P}$ can execute a communication has a continuation where \mathbf{A} is involved;
- S is *lock-free* when in all computations starting from a reachable configuration where a participant $\mathbf{A} \in \mathcal{P}$ can execute, \mathbf{A} is involved;
- S is *deadlock-free* if in none of its reachable configurations s without outgoing transitions there exists $\mathbf{A} \in \mathcal{P}$ willing to communicate.

A system of CFSMs $S = (M_x)_{x \in \mathcal{P}}$ is abstractly represented by the system $\hat{S} = (\mathbb{L}(M_x))_{x \in \mathcal{P}}$. It is the case that lock-freedom, strong lock-freedom, and deadlock-freedom of \hat{S} (in the sense of Def. 4.1) respectively imply liveness, lock-freedom, and deadlock-freedom of S (cf. [7, Prop. 8.12]).

The conditions on c -automata devised in [6] in order to guarantee the above communication properties in the synchronous case turned out to be flawed. This is shown in [7, Sec. 8.3] (cf. [7, Ex. 8.10]).

Fortunately, the conditions given in the present paper can be applied also in the setting of c -automata. As shown in [7, Sec. 8.4], CUI and branch-awareness are decidable.

8 Concluding Remarks

We developed a general and abstract theory of choreographies based on formal languages, in which we recasted known properties and constructions such as projections from global to local specifications. We briefly recap our main contributions, synoptically depicted in Fig. 1.

One of our contributions is the characterisation of systems' correctness in terms of closure under unknown information (CUI). Other communication properties can be ensured by additionally requiring branch awareness (BA).

Finally, the versatility of FCL allows us to capture existing models. We considered two models chosen according to their "proximity" to FCL. The first model, the variant of MPSTs presented in [37], being based on behavioural

types, radically differs from FCL. The second framework, the c-automata in [6], is closer to FCL given that it retraces the connection between automata and formal language theories.

Related work The use of formal language theories for the modelling of concurrent systems dates back to the theory of traces [33]. A trace is an equivalence class of words that differ only for swaps of independent symbols. Closure under concurrency corresponds on finite words to form traces, as we noted after Def. 2.8. An extensive literature has explored a notion of realisability whereby a language of traces is realisable if it is accepted by some class of finite-state automata. Relevant results in this respect are the characterisations in [39,16] (and the optimisation in [22]) for finite words and the ones in [19,20,21] for infinite ones. A key difference of our framework w.r.t. this line of work is that we aim to stricter notions of realisability: in our context it is not enough that the runs of the language may be faithfully executed by a certain class of finite-state automata. Rather we are interested in identifying conditions on the g-languages that guarantee well-behaved executions in “natural” realisations.

Other abstract models of choreographies, e.g. [18,6], have some relation with ours. Conversation protocols (CP) [18], probably the first automata-based model of choreographies, are non-deterministic Büchi automata whose alphabet resembles a constrained variant of our Σ_{int} . A comparison with the g-languages accepted by CPs is not immediate as CPs are based on asynchronous communications (although some connections are evident as noted below Def. 2.6).

Other proposals ascribable to choreographic settings (cf. [25]) define global views that can be seen as g-languages. We focus on synchronous approaches because our current theory needs to be extended to cope with asynchrony.

In [12,31] the correctness of implementations of choreographies (called *choreography conformance*) is studied in a process algebraic setting. The other communication properties we consider here are not discussed there.

The notion of choreography implementation in [12] corresponds to our correctness plus a form of existential termination. It is shown that one can decide whether a system is an implementation of a given choreography, since both languages are generated by finite-state automata, hence language inclusion and existential termination are decidable.

In [31] three syntactic conditions (connectedness, unique points of choice and causality safety) ensure bisimilarity (hence trace equivalence) between a choreography and its projection. Connectedness rules out systems which are not c-closed, while we conjecture that unique points of choice and connectedness together imply our CUI and BA. Causality safety, immaterial in our case, is needed in [31] due to explicit parallel composition.

Many multiparty session type systems [25] have two levels of types (global and local) and one implementation level (local processes). This is the case also for synchronous session type systems such as [30,15]. Our approach, like the session type systems in [37,5], considers only (two) abstract descriptions, g-languages and l-languages. The literature offers several behavioural types featuring correctness-by-construction principles through conditions (known as pro-

jectability or well-branchedness) more demanding than ours. For instance, relations similar to those in Section 6 can be devised for close formalisms, such as [5] whose notion of projection is more general than the one in [37], yet its notion of projectability still implies CUI and BA.

There is a connection between CUI and the closure property CC2 [3] on message-sequence charts (MSCs) [26]. On finite words CC2 and CUI coincide. Actually, CUI can be regarded as a step-by-step way to ensure CC2 on finite words. The relations between our properties and CC3, also used in MSCs, are still under scrutiny.

Future work Our investigation proposes a new point of view for choreography formalisms and the related constructions. As such, a number of extensions and improvements need to be analysed, to check how they may fit in our setting. We list below the most relevant.

First, we need to extend our theory to cope with *asynchronous* communications. While the general approach should apply, it is not immediate how to extend CUI in order to characterize correctness for an asynchronous semantics. This is somehow confirmed by the results in [1,2] on the realisability of MSCs showing that in the asynchronous setting this is a challenging problem.

A second direction is analysing how to drop prefix-closure, so allowing for specifications where the system (and single participants) may stop their execution at some points but not at others; a word would hence represent a complete computation, not only a partial one.

A further direction would unveil the correspondence between closure properties and subtyping relations used in many multiparty session types.

References

1. Rajeev Alur. The benefits of exposing calls and returns. In Martín Abadi and Luca de Alfaro, editors, *CONCUR 2005 - Concurrency Theory, 16th International Conference, CONCUR 2005, San Francisco, CA, USA, August 23-26, 2005, Proceedings*, volume 3653 of *Lecture Notes in Computer Science*, pages 2–3. Springer, 2005. doi:10.1007/11539452_2.
2. Rajeev Alur, Kousha Etessami, and Mihalis Yannakakis. Realizability and verification of MSC graphs. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, *Automata, Languages and Programming, 28th International Colloquium, ICALP 2001, Crete, Greece, July 8-12, 2001, Proceedings*, volume 2076 of *Lecture Notes in Computer Science*, pages 797–808. Springer, 2001. doi:10.1007/3-540-48224-5_65.
3. Rajeev Alur, Kousha Etessami, and Mihalis Yannakakis. Inference of Message Sequence Charts. *IEEE Trans. Software Eng.*, 29(7):623–633, 2003.
4. Jean-Michel Autebert, Joffroy Beauquier, and Luc Boasson. Langages sur des alphabets infinis. *Discrete Applied Mathematics*, 2(1):1 – 20, 1980. URL: <http://www.sciencedirect.com/science/article/pii/0166218X80900505>, doi:10.1016/0166-218X(80)90050-5.
5. Franco Barbanera, Mariangiola Dezani-Ciancaglini, Ivan Lanese, and Emilio Tuosto. Composition and decomposition of multiparty sessions. *Journal of Logical and Algebraic Methods in Programming*, 119:100620, 2021. URL: <http://>

axp@six

- [//www.sciencedirect.com/science/article/pii/S235222082030105X](http://www.sciencedirect.com/science/article/pii/S235222082030105X), doi:10.1016/j.jlamp.2020.100620.
6. 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. doi:10.1007/978-3-030-50029-0_6.
 7. Franco Barbanera, Ivan Lanese, and Emilio Tuosto. Formal choreographic languages (extended version). Technical report, GSSI, 2022. Available at <https://emwww.github.io/home/tr/fcl.pdf>.
 8. Samik Basu and Tevfik Bultan. Choreography conformance via synchronizability. In Sadagopan Srinivasan, Krithi Ramamritham, Arun Kumar, M. P. Ravindra, Elisa Bertino, and Ravi Kumar, editors, *Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28 - April 1, 2011*, pages 795–804. ACM, 2011. doi:10.1145/1963405.1963516.
 9. Samik Basu, Tevfik Bultan, and Meriem Ouederni. Deciding choreography realizability. In *Proceedings of the 39th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2012, Philadelphia, Pennsylvania, USA, January 22-28, 2012*, pages 191–202, 2012. doi:10.1145/2103656.2103680.
 10. Jonas Bonér. *Reactive Microsystems - The Evolution Of Microservices At Scale*. O’Reilly, 2018.
 11. Daniel Brand and Pitro Zafiropulo. On communicating finite-state machines. *J. ACM*, 30(2):323–342, 1983.
 12. Mario Bravetti and Gianluigi Zavattaro. Towards a unifying theory for choreography conformance and contract compliance. In Markus Lumpe and Wim Vanderperren, editors, *Software Composition, 6th International Symposium, SC 2007, Braga, Portugal, March 24-25, 2007, Revised Selected Papers*, volume 4829 of *Lecture Notes in Computer Science*, pages 34–50. Springer, 2007.
 13. Marco Carbone, Kohei Honda, and Nobuko Yoshida. Structured communication-centered programming for web services. *ACM Trans. Program. Lang. Syst.*, 34(2):8:1–8:78, 2012. doi:10.1145/2220365.2220367.
 14. 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.
 15. Mariangiola Dezani-Ciancaglini, Silvia Ghilezan, Svetlana Jaksic, Jovanka Pantovic, and Nobuko Yoshida. Precise subtyping for synchronous multiparty sessions. In Simon Gay and Jade Alglave, editors, *Proceedings Eighth International Workshop on Programming Language Approaches to Concurrency- and Communication-Entric Software, PLACES 2015, London, UK, 18th April 2015*, volume 203 of *EPTCS*, pages 29–43, 2015. doi:10.4204/EPTCS.203.3.
 16. Christine Duboc. Mixed product and asynchronous automata. *TCS*, 48(3):183–199, 1986. doi:10.1016/0304-3975(86)90094-0.
 17. Samuel Eilenberg. *Automata, languages, and machines., B*. Pure and applied mathematics. Academic Press, 1976. URL: <https://www.worldcat.org/oclc/310535259>.
 18. Xiang Fu, Tevfik Bultan, and Jianwen Su. Conversation protocols: a formalism for specification and verification of reactive electronic services. *TCS*, 328(1-2):19–37, 2004.

19. Paul Gastin. Infinite traces. In Irène Guessarian, editor, *Semantics of Systems of Concurrent Processes, LITP Spring School on Theoretical Computer Science, La Roche Posay, France, April 23-27, 1990, Proceedings*, volume 469 of *Lecture Notes in Computer Science*, pages 277–308. Springer, 1990. doi:10.1007/3-540-53479-2_12.
20. Paul Gastin. Recognizable and rational languages of finite and infinite traces. In Christian Choffrut and Matthias Jantzen, editors, *STACS 91, 8th Annual Symposium on Theoretical Aspects of Computer Science, Hamburg, Germany, February 14-16, 1991, Proceedings*, volume 480 of *Lecture Notes in Computer Science*, pages 89–104. Springer, 1991. doi:10.1007/BFb0020790.
21. Paul Gastin, Antoine Petit, and Wieslaw Zielonka. A kleene theorem for infinite trace languages. In Javier Leach Albert, Burkhard Monien, and Mario Rodríguez-Artalejo, editors, *Automata, Languages and Programming, 18th International Colloquium, ICALP91, Madrid, Spain, July 8-12, 1991, Proceedings*, volume 510 of *Lecture Notes in Computer Science*, pages 254–266. Springer, 1991. doi:10.1007/3-540-54233-7_139.
22. Blaise Genest and Anca Muscholl. Constructing exponential-size deterministic zielonka automata. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, volume 4052 of *Lecture Notes in Computer Science*, pages 565–576. Springer, 2006. doi:10.1007/11787006_48.
23. Silvia Ghilezan, Svetlana Jaksic, Jovanka Pantovic, Alceste Scalas, and Nobuko Yoshida. Precise subtyping for synchronous multiparty sessions. *J. Log. Algebraic Methods Program.*, 104:127–173, 2019. doi:10.1016/j.jlamp.2018.12.002.
24. Kohei Honda, Nobuko Yoshida, and Marco Carbone. Multiparty asynchronous session types. *Journal of the ACM*, 63(1):9:1–9:67, 2016. Extended version of a paper presented at POPL08. doi:10.1145/2827695.
25. Hans Hüttel, Ivan Lanese, Vasco T. Vasconcelos, Luís Caires, Marco Carbone, Pierre-Malo Deniérou, Dimitris Mostros, 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.
26. ITU Telecommunication Standardization Sector. *ITU-T recommendation Z.120. Message Sequence Charts (MSC'96)*, 1996.
27. Nickolas Kavantzias, David Burdett, Gregory Ritzinger, Tony Fletcher, Yves Lafon, and Charlton Barreto. Web services choreography description language version 1.0. Technical report, W3C, 2005. <http://www.w3.org/TR/ws-cdl-10/>.
28. Naoki Kobayashi. A Type System for Lock-Free Processes. *Information and Computation*, 177:122–159, 2002.
29. Naoki Kobayashi and Davide Sangiorgi. A hybrid type system for lock-freedom of mobile processes. *ACM Trans. Program. Lang. Syst.*, 32(5):16:1–16:49, 2010. doi:10.1145/1745312.1745313.
30. Dimitrios Kouzapas and Nobuko Yoshida. Globally governed session semantics. *Log. Methods Comput. Sci.*, 10(4), 2014. doi:10.2168/LMCS-10(4:20)2014.
31. Ivan Lanese, Claudio Guidi, Fabrizio Montesi, and Gianluigi Zavattaro. Bridging the gap between interaction- and process-oriented choreographies. In *Software Engineering and Formal Methods, SEFM 2008*, pages 323–332, 2008.
32. Julien Lange, Nicholas Ng, Bernardo Toninho, and Nobuko Yoshida. Fencing off go: liveness and safety for channel-based programming. In Giuseppe Castagna and Andrew D. Gordon, editors, *Proceedings of the 44th ACM SIGPLAN Symposium*

- on *Principles of Programming Languages, POPL 2017, Paris, France, January 18-20, 2017*, pages 748–761. ACM, 2017. URL: <http://dl.acm.org/citation.cfm?id=3009847>.
33. Antoni W. Mazurkiewicz. Trace theory. In Wilfried Brauer, Wolfgang Reisig, and Grzegorz Rozenberg, editors, *Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986, Part II, Proceedings of an Advanced Course, Bad Honnef, Germany, 8-19 September 1986*, volume 255 of *Lecture Notes in Computer Science*, pages 279–324. Springer, 1986. doi:10.1007/3-540-17906-2_30.
 34. OMG. *Business Process Model and Notation (BPMN), Version 2.0*, January 2011. <https://www.omg.org/spec/BPMN>.
 35. Luca Padovani. From lock freedom to progress using session types. In Nobuko Yoshida and Wim Vanderbauwhede, editors, *Proceedings 6th Workshop on Programming Language Approaches to Concurrency and Communication-cEntric Software, PLACES 2013, Rome, Italy, 23rd March 2013*, volume 137 of *EPTCS*, pages 3–19, 2013. doi:10.4204/EPTCS.137.2.
 36. Roman R. Redziejowski. Infinite-word languages and continuous mappings. *TCS*, 43:59–79, 1986. doi:10.1016/0304-3975(86)90166-0.
 37. Paula Severi and Mariangiola Dezani-Ciancaglini. Observational equivalence for multiparty sessions. *Fundam. Informaticae*, 170(1-3):267–305, 2019. doi:10.3233/FI-2019-1863.
 38. Ludwig Staiger. ω -languages. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages, Volume 3: Beyond Words*, pages 339–387. Springer, 1997. doi:10.1007/978-3-642-59126-6_6.
 39. Wieslaw Zielonka. Notes on finite asynchronous automata. *RAIRO Theor. Informatics Appl.*, 21(2):99–135, 1987. doi:10.1051/ita/1987210200991.