Towards Refinable Choreographies*

Ugo de'Liguoro
Università di Torino, Italy

Hernán Melgratti
ICC
Universidad de Buenos Aires
Conicet, Argentina

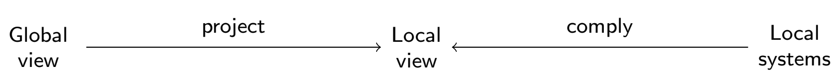
Emilio Tuosto
School of Informatics, Leicester, UK
GSSI, L'Aquila, Italy

We investigate refinement in the context of choreographies. We introduce refinable global choreographies allowing for the underspecification of protocols, whose interactions can be refined into actual protocols. Arbitrary refinements may spoil *well-formedness*, that is the sufficient conditions that guarantee a protocol to be implementable. We introduce a typing discipline that enforces well-formedness of typed choreographies. Then we unveil the relation among refinable choregraphies and their admissible refinements in terms of an axiom scheme.

1 Introduction

The advent of structured programming [7] is probably behind the widespread use of refinement methods in computer science. Refinement is paramount in many formal methods, in software engineering, and in verification, because the possibility of structuring a system into simpler components is crucial to tackle the complexity of a system.

In this paper we investigate the refinement of choreographies of message-passing systems. In this domain, a choreography specifies the coordination of distributed components (aka *participants* or *roles*) by disciplining the exchange of messages. Following W3C [11], we envisage a choreography as a contract consisting of a *global view* that can be used as a blueprint for defining each participant. A global view is basically an *application-level* protocol realised through the coordination of the resulting *local views*, the specifications of participants. This description is the ground for the so-called top-down engineering represented by the following diagram:



(1)

where the 'projection' operation produces local views from the global ones and the operation 'comply' verifies that the behaviour of each participant adheres to the one of the corresponding local view.

Choreographic approaches are appealing because, unlike orchestration, they do not require an explicit coordinator (see [2] for a deeper discussion). Moreover, global views allow developers to work independently on different components.

Despite the main advantages discussed above, choreographic approaches suffer a main drawback: the lack of support for modular development. This shortcoming is present in standards such as BPMN

* Research partly supported by the EU H2020 RISE programme under the Marie Skodowska-Curie grant agreement No 778233 . Work partially funded by MIUR project PRIN 2017FTXR7S *IT MATTERS* (Methods and Tools for Trustworthy Smart Systems) . Research partially supported by the UBACyT projects 20020170100544BA and 20020170100086BA, and by the PIP project 11220130100148CO. COST Action: EUTypes CA15123; and local funds of the University of Turin: Ricerca locale Linea A (BERS-RILO-17-03) - Fondazioni logiche della computazione, Ricerca locale Linea A (PAOL-RILO-18-01) - Fondazioni logiche della computazione, Ricerca locale 2019 Linea A (DE-U-RILO-19-01) - Logica della computazione.
The authors thank the anonymous reviewers for their comments and the interesting discussions on the forum of ICE20.

J. Lange, A. Mavridou, L. Safina, A. Scalas (Eds.):
13th Interaction and Concurrency Experience (ICE 2020).
EPTCS 324, 2020, pp. 61–77, doi:10.4204/EPTCS.324.6

or in workflow patterns and languages [3] and it has been more recently flagged also for choreographic programming [4].

We propose a choreographic model of message-passing applications based on point-to-point communication equipped with a simple refinement mechanism. Let us illustrate this through some simple examples. This gives us the opportunity to informally use *global choreographies* [14, 9] (g-choreographies, for short), the formalisation of global views adopted here for the technical development of the paper.

Consider the g-choreography

$$\mathsf{C} \overset{\mathsf{md}}{\dashrightarrow} \mathsf{S} + \mathsf{C} \overset{\mathsf{req}}{\dashrightarrow} \mathsf{S}; \mathsf{S} \overset{\mathsf{done}}{\dashrightarrow} \mathsf{C} \tag{2}$$

where a client C either sends some meta-data md or a request req to a server S. In the former case the protocol terminates, while in the latter the server is supposed to send back a response done to C. The dashed arrows above represent *refinable interactions*, that is interactions that can be replaced so to refine the application-level protocol. For instance, to allow S to send C some statistical information in the second branch of (2) we can refine $\mathsf{S} \overset{\mathsf{done}}{\dashrightarrow} \mathsf{C}$ with $\mathsf{S} \xrightarrow{\mathsf{stats}} \mathsf{C}; \mathsf{S} \xrightarrow{\mathsf{done}} \mathsf{C}$ and obtain

$$\mathsf{C} \overset{\mathsf{md}}{\dashrightarrow} \mathsf{S} + \mathsf{C} \overset{\mathsf{req}}{\dashrightarrow} \mathsf{S}; \mathsf{S} \xrightarrow{\mathsf{stats}} \mathsf{C}; \mathsf{S} \xrightarrow{\mathsf{done}} \mathsf{C} \tag{3}$$

where the interactions with the solid arrow are now "ground", namely they cannot be further refined.

This is our simple refinement mechanism: replace a refinable interaction with a more complex (refinable) protocol. A key goal here is to provide a mechanism of refinement without spoiling *well-formedness* conditions. Basically, well-formedness conditions ensure that the application-level protocol modelled by the global view is faithfully executed by the participants that comply with the projected local views. Let us again explain this with an example. Suppose we refine (2) by replacing each refinable interaction with its ground version but for $\mathsf{C} \overset{\mathsf{md}}{\dashrightarrow} \mathsf{S}$, which is replaced by $\mathsf{C} \xrightarrow{\mathsf{md}} \mathsf{B}; \mathsf{B} \xrightarrow{\mathsf{md}} \mathsf{S}$, where B is a brokerage service mediating the exchange of md. We obtain

$$\mathsf{C} \xrightarrow{\mathsf{md}} \mathsf{B}; \mathsf{B} \xrightarrow{\mathsf{md}} \mathsf{S} + \mathsf{C} \xrightarrow{\mathsf{req}} \mathsf{S}; \mathsf{S} \xrightarrow{\mathsf{done}} \mathsf{C} \tag{4}$$

The g-choreography above is not well-formed because the broker B is oblivious of the second branch. Namely, B will be stuck waiting for message md should C opt for the second branch of the choice.

Contributions & Structure We introduce a simple mechanism for refining global views of choreographies. Firstly, we equip an existing formal language expressing *global choreographies* (g-choreographies, for short) with a semantics based on event structures (surveyed in Section 2) and identifying a typing discipline (Section 3) that checks sufficient conditions for well-formedness. Secondly, we extend g-choreographies with our refinement mechanism (Section 4). A key design choice of our framework is to ground refinements on the concept *refinable interactions*. Inspired by the action refinement mechanism of process algebra, we consider refinable g-choreographies those where refinable interactions may occur. Refinable g-choreographies play the role of incomplete specifications where, by repeated replacements of refinable interactions, one can incrementally attain a fully specified global view.

One problem that may arise in this process is that refinements could spoil well-formedness and hence compromise realisability of global views. To avoid this we extend the typing discipline for non-refinable g-choreographies to refinable ones and show that the replacement of a refinable interaction with a g-choreography typable with the same type ensures realisability.

We discuss related work and draw some conclusions in Section 5.

2 Background

We recall basic notions of *event structures* used in Section 3 to give semantics to refinable choreographies. Event structures model concurrency in terms of partial orders of labelled events. We focus our attention on event structures of communication events. Let $\mathscr{P}$ be a set of *participants* (ranged over by A, B, etc.) and $\mathscr{M}$ be a set of *messages* (ranged over by m, x, etc.). We take $\mathscr{P}$ and $\mathscr{M}$ disjoint. Let

$$\mathscr{C} = (\mathscr{P} \times \mathscr{P}) \setminus \{(\mathsf{A},\mathsf{A}) \mid \mathsf{A} \in \mathscr{P}\} \qquad \mathscr{L}^! = \mathscr{C} \times \{!\} \times \mathscr{M} \qquad \mathscr{L}^? = \mathscr{C} \times \{?\} \times \mathscr{M}$$

be the sets of *channels*, output labels, and input labels respectively. We write $\mathsf{A}\,\mathsf{B}!\mathsf{m} \in \mathscr{L}^!$ and $\mathsf{A}\,\mathsf{B}?\mathsf{m} \in \mathscr{L}^?$ instead of $((\mathsf{A},\mathsf{B}),!,\mathsf{m}) \in \mathscr{L}^!$ and $((\mathsf{A},\mathsf{B}),?,\mathsf{m}) \in \mathscr{L}^?$. The *subject* of a label l, written $\mathsf{sbj}\ l$, is defined as $\mathsf{sbj}\ \mathsf{A}\,\mathsf{B}!\mathsf{m} = \mathsf{sbj}\ \mathsf{B}\,\mathsf{A}?\mathsf{m} = \{\mathsf{A}\}$. The elements of $\mathscr{L} = \mathscr{L}^! \cup \mathscr{L}^?$ (ranged over by l) will be used to label the events of our event structures. The *co-action* of $l \in \mathscr{L}$ is defined as $co(\mathsf{A}\,\mathsf{B}!\mathsf{m}) = \mathsf{A}\,\mathsf{B}?\mathsf{m}$ and $co(\mathsf{A}\,\mathsf{B}?\mathsf{m}) = \mathsf{A}\,\mathsf{B}!\mathsf{m}$ and extends element-wise on sets of actions.

Definition 1 (Event structures)**.** *An* event structure *labelled over* $\mathscr{L}$ *(shortly* event structure*) is a tuple* $\mathscr{E} = (E, \leq, \#, \lambda)$ *where*

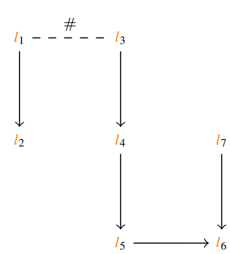
- E *a set of* events
- $\leq\ \subseteq E \times E$ *a partial order, the* causality *relation*
- $\#\ \subseteq E \times E$ *a symmetric and irreflexive relation, the* conflict *relation,*
- $\lambda : E \to \mathscr{L}$ *a* labelling *mapping.*

are such that

- *each event has only finitely many predecessors, namely* $\forall e \in E.\ \{e' \in E \mid e' \leq e\}$ *is finite, and*
- *conflicts are hereditary, namely* $\forall e, e', e'' \in E.\ e \# e'\ \ \&\ \ e' \leq e'' \implies e \# e''$

If $\mathscr{E} = (E, \leq, \#, \lambda)$ *is an event structure then* $\min(\mathscr{E}), \max(\mathscr{E}) \subseteq E$ *are the minimal and the maximal elements in the poset* $(E, \leq)$*. We define* $\varepsilon = (\emptyset, \emptyset, \emptyset, \emptyset)$ *as the empty event structure, where* $\lambda_\emptyset = \emptyset$ *is the empty mapping.*

Notice that if $\mathscr{E} \neq \varepsilon$ then minimal elements do exist, while this is not necessarily the case for maximal ones. We depict event structures following the customary representation of the literature [12] as the diagram (5) below; instead of events though, we prefer to use their labels, for instance:



(5)

represents an event structure with events $e_1, \ldots, e_7$ (not represented in the diagram above) and labelled respectively by $l_1, \ldots, l_7$ where

- the event e_1 precedes e_2 (i.e., $e_1 \leq e_2$ in the partial order of the event structure)
- events e_1 and e_3 are in conflict; recall that the conflict relation is hereditary, hence e_1 and e_2 are in conflict with all other events but e_7

- events e_2 and e_6 are maximal; the latter follows both e_5 and e_7
- events e_5 and e_7 are independent of each other (actually, e_7 is independent of any event but e_6).

In our diagrams we adopt the implicit assumption that each occurrence of a label correspond to a different event; for instance, in the diagram (5), even if two labels, say l_1 and l_7 were equal, the corresponding events would be distinct (i.e., $e_1 \neq e_7$).

An event structure induces a natural order and conflict relations on the events performed by each participant. More precisely, the *projection* of an event structure $\mathscr{E} = (E, \leq, \#, \lambda)$ on a participant $\mathsf{A} \in \mathscr{P}$ is the structure

$$\mathscr{E} \restriction \mathsf{A} = (E \restriction \mathsf{A}, \leq\restriction \mathsf{A}, \# \restriction \mathsf{A}, \lambda \restriction \mathsf{A})$$

where

$E \restriction \mathsf{A} = \{e \in E \mid \mathsf{sbj}\ \lambda(e) = \mathsf{A}\}$

$\leq\restriction \mathsf{A} = \leq \cap (E \restriction \mathsf{A})^2$ and $\# \restriction \mathsf{A} = \# \cap (E \restriction \mathsf{A})^2$

$\lambda \restriction \mathsf{A} = \lambda\big|_{E \restriction \mathsf{A}}$, namely the restriction of λ to $E \restriction \mathsf{A}$

Trivially, the induced relations form an event structure.

Lemma 1. *If $\mathscr{E}$ is an event structure and $\mathsf{A} \in \mathscr{P}$ then $\mathscr{E} \restriction \mathsf{A}$ is an event structure.*

Proof. Immediate since $\leq\restriction \mathsf{A} \subseteq \leq$ and $\# \restriction \mathsf{A} \subseteq \#$. □

We now define a few operations instrumental to our technical development. Let $\mathscr{E}_0 = (E_0, \leq_0, \#_0, \lambda_0)$ and $\mathscr{E}_1 = (E_1, \leq_1, \#_1, \lambda_1)$ be labelled event structures.

The product operation $\_ \otimes \_$ yields the disjoint union of event structures preserving their orders, conflicts, and labellings; it is define as

$$\mathscr{E}_0 \otimes \mathscr{E}_1 = (E_0 \uplus E_1, \leq, \#, \lambda)$$

where writing $\iota_i : E_i \to E_0 \uplus E_1$ for the injections, we set

$$\iota_i e \leq \iota_j e' \iff i = j \ \&\ e \leq_i e' \qquad \iota_i e \# \iota_j e' \iff i = j \ \&\ e \#_i e' \qquad \lambda(\iota_i e) = \lambda_i(e)$$

The sum $\sum_{i \in I} \mathscr{E}_i$ yields the disjoint union of a family $\{\mathscr{E}_i\}_{i \in I}$ of event structures $\mathscr{E}_i = (E_i, \leq_i, \#_i, \lambda_i)$ preserving their orders and labellings while introducing conflicts among events of different members of the family; it is defined as the event structure $(\biguplus_{i \in I} E_i, \leq, \#, \lambda)$ where, writing $\iota_i : E_i \to \biguplus_{i \in I} E_i$ for the injections, the following hold:

$$\iota_i e \leq \iota_j e' \iff i = j \ \&\ e \leq_i e' \qquad \iota_i e \# \iota_j e' \iff i \neq j \vee (i = j \ \&\ e \#_i e') \qquad \lambda(\iota_i e) = \lambda_i(e)$$

In particular we write

$$\mathscr{E}_0 + \mathscr{E}_1 = \sum_{i \in \{0,1\}} \mathscr{E}_i \qquad \text{and} \qquad \sum_{i \in I} \mathscr{E} = \sum_{i \in I} \mathscr{E}_i \quad \text{where } \mathscr{E}_i = \mathscr{E} \text{ for all } i \in I$$

Lemma 2. *If $\mathscr{E}_0, \mathscr{E}_1$ are event structures and $\{\mathscr{E}_i\}_{i \in I}$ is a family of event structures then*

$$\mathscr{E}_0 \otimes \mathscr{E}_1 \qquad \textit{and} \qquad \sum_{i \in I} \mathscr{E}_i \qquad \textit{are event structures.}$$

Definition 2 (Configuration domain). *If $\mathscr{E} = (E, \leq, \#, \lambda)$ is an event structure a set of events $x \subseteq E$ is a* configuration *if*

1. $e \leq e'$ & $e' \in x \Longrightarrow e \in x$ *(x is downward closed)*
2. $\forall e, e' \in x.\ \neg(e \# e')$ *(x is consistent)*

Let $C = \{x \subseteq E \mid x \text{ a configuration}\}$; the domain of configurations *of $\mathscr{E}$ is the poset $\mathscr{D}(\mathscr{E}) = (C, \subseteq)$. We say that $x \in C$ is* maximal *if it is such in $\mathscr{D}(\mathscr{E})$: $\mathscr{C}_{\max}(\mathscr{E})$ is the set of maximal configurations.*

Being conflict-free and maximal, configurations in $\mathscr{C}_{\max}(\mathscr{E})$ correspond to branches of events of $\mathscr{E}$.

3 Well-formedness by Typing

We formalise global views of choreographies as g-choreographies [9, 14]. Although we maintain the original syntax, we provide a new semantics of g-choreographies based on event structures. This is instrumental to identify a simple notion of well-formedness that can be statically checked.

3.1 Global Choreographies

Definition 3 introduces *global choreographies*. The syntax of a g-choreography is given by the grammar below that we borrow from [14].

Definition 3 (Global Choreographies). *The set $\mathscr{G}$ of* global choreographies (g-choreographies for short) *consists of the terms* G *derived by the grammar*

$$\begin{array}{rlr} \mathsf{G} ::= & \mathbf{0} & \textit{empty} \quad (6) \\ \mid & \mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B} & \textit{interaction} \quad (7) \\ \mid & \mathsf{G}; \mathsf{G}' & \textit{sequential} \quad (8) \\ \mid & \mathsf{G} \mid \mathsf{G}' & \textit{parallel} \quad (9) \\ \mid & \mathsf{G} + \mathsf{G}' & \textit{choice} \quad (10) \end{array}$$

such that $\mathsf{A} \neq \mathsf{B}$ in interactions (7) *We let $\mathscr{P}(\mathsf{G})$ be the set of participants occurring in* G.

Besides the empty choreography $\mathbf{0}$, the syntax of Definition 3 allows us to specify choreographies whose basic elements are interactions $\mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B}$ which represent that participant A sends message m to participant B, which in turn should receive it. Finally, g-choreographies can be composed sequentially, in parallel, and non-deterministically. The syntax in [14] encompasses iterative g-choreographies which we drop for simplicity. Adding iteration can be done following standard techniques at the cost of a substantial increase of the technical complexity.

Example 3. The term in (4) in Section 1 is a g-choreography. ◇

We now give the semantics of g-choreographies in terms of event structures. To this purpose, note that not every G is "meaningful" because G can specify protocols where the behaviour of some participants, say B, depends on choices made by others that are not properly propagated to B. The following example illustrates this.

Example 4. The g-choreography $\mathsf{G} = \mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{C}; \mathsf{B} \xrightarrow{\mathsf{m}} \mathsf{C} + \mathsf{A} \xrightarrow{\mathsf{n}} \mathsf{C}; \mathsf{B} \xrightarrow{\mathsf{n}} \mathsf{C}$ specifies a protocol where A decides whether to send m or n to C. In either case B should mimic A and send the same message to C. However, in a distributed implementation of this protocol B is oblivious of the decision of A; hence, e.g., B could send message m while A decided to send message n. ◇

To mitigate the problem above, we give *well-formedness* conditions that rule out meaningless g-choreographies. We start with *well-branchedness*.

Definition 4 (Well-branchedness)**.** *Event structures $\mathscr{E}_0$ and $\mathscr{E}_1$ are* well-branched *(in symbols $wb(\mathscr{E}_0, \mathscr{E}_1)$) if, for $\mathscr{E} = \mathscr{E}_0 + \mathscr{E}_1 = (E, \leq, \#, \lambda)$, the following two conditions hold:*

$$\begin{array}{rl} \textit{determined choice:} & \forall \mathsf{B} \in \mathscr{P} : \ (\mathscr{E}_0 \upharpoonright \mathsf{B} = \varepsilon \iff \mathscr{E}_1 \upharpoonright \mathsf{B} = \varepsilon) \ \& \\ & \qquad \forall e, e' \in \min(E \upharpoonright \mathsf{B}) : \ e \# e' \implies \lambda(e) \neq \lambda(e') \\ \textit{unique selector:} & \exists \mathsf{A} \in \mathscr{P} : \ \emptyset \neq \lambda(\min(E \upharpoonright \mathsf{A})) \subseteq \{ l \in \mathscr{L}^! \mid \mathit{sbj}\ l = \mathsf{A} \} \ \& \\ & \qquad \forall \mathsf{B} \neq \mathsf{A} \in \mathscr{P} : \ \lambda(\min(E \upharpoonright \mathsf{B})) \subseteq \{ l \in \mathscr{L}^? \mid \mathit{sbj}\ l = \mathsf{B} \} \end{array}$$

We dub active *the unique participant* A *satisfying the second condition and* passive *the others.*

Well-branchedness is akin to the conditions on behavioural types that enforce choice determinacy. Namely, each choice is determined by a unique participant, dubbed selector, which starts to send messages to the others and that any non-selector participant becomes aware of the choice taken by the selector just because of the messages received on a branch.

We re-cast the notion of *well-forkedness* in [14] in terms of event structures.

Definition 5 (Well-forkedness)**.** *Two event structures $\mathscr{E} = (E, \leq, \#, \lambda)$ and $\mathscr{E}' = (E', \leq, \#', \lambda')$ are* well-forked *(in symbols $wf(\mathscr{E}, \mathscr{E}')$) if $\lambda(E) \cap \lambda'(E') = \emptyset$.*

As well-branchedness, the parallel composition of g-choreographies is subject to some conditions. As observed in [14], "confusion" may arise when different threads of participants exchange the same message: the message meant to be received by a thread is received by the other. If this happens there is a violation of the causal order of the events. The next example illustrates the problem.

Example 5. The g-choreography $\mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B}; \mathsf{B} \xrightarrow{\mathsf{m}} \mathsf{C} \mid \mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B}; \mathsf{B} \xrightarrow{\mathsf{n}} \mathsf{D}$ is not well-forked because the interaction between B and C should start *after* the "left thread" of A had sent message m. However, it could happen that the left thread of B receives the message sent by the right thread of A so violating the specification. And likewise for the "right threads". ⋄

Definition 6 (Sequential composition)**.** *Let $\mathscr{E}$, $\mathscr{E}'$ be event structures and*

$$(E'', \leq'', \#'', \lambda'') = \mathscr{E} \otimes \sum_{x \in \mathscr{C}_{\max}(\mathscr{E})} \mathscr{E}'_x$$

where the structures $\mathscr{E}'_x = (E_x, \leq_x, \#_x, \lambda_x)$ are disjoint copies of $\mathscr{E}'$, then

$$\mathsf{seq}(\mathscr{E}, \mathscr{E}') = (E'', \leq'' \cup \bigcup_{x \in \mathscr{C}_{\max}(\mathscr{E})} \{ (e, e') \in x \times E_x \mid \mathit{sbj}\ \lambda''(e) = \mathit{sbj}\ \lambda''(e') \}, \#'', \lambda'').$$

The intuition of the definition of $\mathsf{seq}(\mathscr{E}, \mathscr{E}')$ is that any branch $x \in \mathscr{C}_{\max}(\mathscr{E})$ of $\mathscr{E}$ is concatenated to a (pairwise incompatible) copy of $\mathscr{E}'_x$, where events in $\mathscr{E}$ cause those of $\mathscr{E}'_x$ with labels having the same subject. Admittedly, in the context of Definition 7 this is unnecessarily abstract, since an event structure $\mathscr{E}$ interpreting a g-choreography is finite, and hence $\mathscr{C}_{\max}(\mathscr{E})$ and any of its elements are such: hence any $x \in \mathscr{C}_{\max}(\mathscr{E})$ includes a finite subset of maximals with respect to $\leq_{\mathscr{E}}$. However the definition applies to infinite structures as well.

Lemma 6. *If $\mathscr{E}, \mathscr{E}'$ are event structures, then $\mathsf{seq}(\mathscr{E}, \mathscr{E}')$ is an event structure.*

We can now give a denotational semantics of g-choreographies. We require our semantics to be defined only on g-choreographies amenable of being realised by distributed components satisfying the following requirements:

- no extra components: each component uniquely corresponds to a participant of the g-choreography
- no extra communications: each communication among the components uniquely corresponds to some communication events of the (semantics of the) g-choreography

These requirements impose that the communication behaviour of a realisation of a g-choreography faithfully reflects the communication events of the g-choreography.

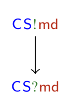
Definition 7 (Semantics)**.** *Let* G *be a g-choreography. The* semantics $[\![\mathsf{G}]\!]$ *of* G *is the partial mapping assigning an event structure to* G *according to the following inductive clauses:*

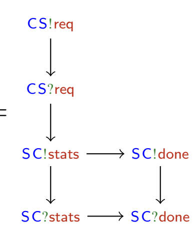
$$\begin{array}{rcl}
[\![\mathbf{0}]\!] & = & \varepsilon \\
[\![\mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B}]\!] & = & (\{e_1, e_2\}, \{e_1 < e_2\}, \emptyset, \{e_1 \mapsto \mathsf{A\,B!m},\ e_2 \mapsto \mathsf{B\,A?m}\}) \\
[\![\mathsf{G};\mathsf{G}']\!] & = & \mathsf{seq}([\![\mathsf{G}]\!], [\![\mathsf{G}']\!]) \\
[\![\mathsf{G} \mid \mathsf{G}']\!] & = & \begin{cases} [\![\mathsf{G}]\!] \otimes [\![\mathsf{G}']\!] & \textit{if } wf([\![\mathsf{G}]\!], [\![\mathsf{G}']\!]) \\ \bot & \textit{otherwise} \end{cases} \\
[\![\mathsf{G} + \mathsf{G}']\!] & = & \begin{cases} [\![\mathsf{G}]\!] + [\![\mathsf{G}']\!] & \textit{if } wb([\![\mathsf{G}]\!], [\![\mathsf{G}']\!]) \\ \bot & \textit{otherwise} \end{cases}
\end{array}$$

where if either $[\![\mathsf{G}]\!]$ *or* $[\![\mathsf{G}']\!]$ *is* $\bot$*, then* $\mathsf{seq}([\![\mathsf{G}]\!], [\![\mathsf{G}']\!])$*,* $[\![\mathsf{G}]\!] \otimes [\![\mathsf{G}']\!]$ *and* $[\![\mathsf{G}]\!] + [\![\mathsf{G}']\!]$ *are all equal to* $\bot$*. Finally we say that* G *is* well-formed *if* $[\![\mathsf{G}]\!] \neq \bot$*.*

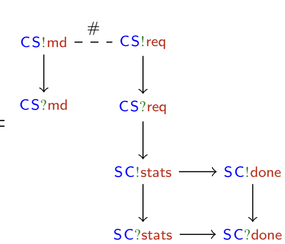
We say that a g-choreography G is *well-formed* when each choice subterm of G is well-branched and each parallel subterm of G is well-forked.

Example 7. Let us spell out the semantics of the g-choreography $\mathsf{G} = \mathsf{C} \xrightarrow{\mathsf{md}} \mathsf{S} + \mathsf{C} \xrightarrow{\mathsf{req}} \mathsf{S}; \mathsf{S} \xrightarrow{\mathsf{stats}} \mathsf{C}; \mathsf{S} \xrightarrow{\mathsf{done}} \mathsf{C}$ obtained by the refinement (3) (cf. Section 1) with the further refinement of $\mathsf{C} \overset{\mathsf{md}}{\dashrightarrow} \mathsf{S}$ with its ground counterpart $\mathsf{C} \xrightarrow{\mathsf{md}} \mathsf{S}$. By Definition 7, $[\![\mathsf{G}]\!]$ is defined if $wb([\![\mathsf{C} \xrightarrow{\mathsf{md}} \mathsf{S}]\!], [\![\mathsf{C} \xrightarrow{\mathsf{req}} \mathsf{S}; \mathsf{S} \xrightarrow{\mathsf{stats}} \mathsf{C}; \mathsf{S} \xrightarrow{\mathsf{done}} \mathsf{C}]\!])$ holds. We now verify that this is the case. By definition, we have:

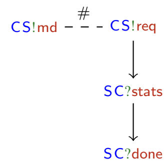
$[\![\mathsf{C} \xrightarrow{\mathsf{md}} \mathsf{S}]\!] =$ 

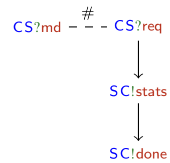
and $[\![\mathsf{C} \xrightarrow{\mathsf{req}} \mathsf{S}; \mathsf{S} \xrightarrow{\mathsf{stats}} \mathsf{C}; \mathsf{S} \xrightarrow{\mathsf{done}} \mathsf{C}]\!] =$ 

The sum operation on event structures introduces conflicts between the events in $[\![\mathsf{C} \xrightarrow{\mathsf{md}} \mathsf{S}]\!]$ and those in $[\![\mathsf{C} \xrightarrow{\mathsf{req}} \mathsf{S}; \mathsf{S} \xrightarrow{\mathsf{stats}} \mathsf{C}; \mathsf{S} \xrightarrow{\mathsf{done}} \mathsf{C}]\!]$, hence:

$\mathscr{E} = [\![\mathsf{C} \xrightarrow{\mathsf{md}} \mathsf{S}]\!] + [\![\mathsf{C} \xrightarrow{\mathsf{req}} \mathsf{S}; \mathsf{S} \xrightarrow{\mathsf{stats}} \mathsf{C}; \mathsf{S} \xrightarrow{\mathsf{done}} \mathsf{C}]\!] =$ 

(recall that conflicts are hereditary, hence it is enough to put only minimal events in conflict). Now we look at the projections on C and on S of $\mathscr{E}$:

$[\![\mathsf{G}]\!] \upharpoonright \mathsf{C} =$ 

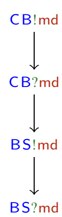
and $[\![\mathsf{G}]\!] \upharpoonright \mathsf{S} =$ 

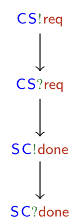
where in $\mathscr{E} \upharpoonright \mathsf{C}$ the minimal events are in conflict because they are in conflict in $\mathscr{E}$ by construction and in $[\![\mathsf{G}]\!] \upharpoonright \mathsf{S}$ they are in conflict because in $\mathscr{E}$ the events inherit previous conflict. It is easy to verify that the conditions of Definition 4 are therefore verified. $\diamond$

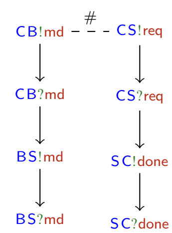
Example 8. Recall the g-choreography (4) in Section 1 which we rewrite as $\mathsf{G}_{err} = \mathsf{G} + \mathsf{G}'$ where

$$\mathsf{G} = \mathsf{C} \xrightarrow{\mathsf{md}} \mathsf{B}; \mathsf{B} \xrightarrow{\mathsf{md}} \mathsf{S} \qquad \text{and} \qquad \mathsf{G}' = \mathsf{C} \xrightarrow{\mathsf{req}} \mathsf{S}; \mathsf{S} \xrightarrow{\mathsf{done}} \mathsf{C}$$

Let us show that $[\![\mathsf{G}_{err}]\!] = \perp$. In fact,

$[\![\mathsf{G}]\!] =$ 

and $[\![\mathsf{G}']\!] =$ 

hence $[\![\mathsf{G}]\!] + [\![\mathsf{G}']\!] =$ 

It is easy to check that the determined choice condition of Definition 4 does not hold for B. $\diamond$

3.2 Typing well-formedness

By Definition 7, non-wellformed g-choreographies G are meaningless, i.e. $[\![\mathsf{G}]\!] = \perp$. However, it is too expensive to check that $[\![\mathsf{G}]\!] \neq \perp$ via a direct inspection of the event structure $[\![\mathsf{G}]\!]$. Also, it is hard to see how to extend the semantics to refinable communications, as their intended meaning is an infinite set of possible realisations by concrete g-choreographies. To circumvent this difficulty we formalise sufficient conditions for well-formedness via a typing system.

Our typing discipline assigns to a g-choreography type $\langle \phi, \Lambda \rangle$ where $\phi \subseteq \mathscr{L}$ and $\Lambda \subseteq \mathscr{L}$. Intuitively, ϕ and Λ are respectively the labels of the first and last events in the g-choreography. Our judgements have the form

$$\Pi \vdash \mathsf{G} : \langle \phi, \Lambda \rangle$$

and their intended meaning is: the g-choreography G has a defined semantics and it has type $\langle \phi, \Lambda \rangle$ under the assumption that its participants are those in $\Pi \subseteq \mathscr{P}$.

Remark 9. *We could avoid the use of the context* Π *in judgements; we however prefer to explicitly list relevant participants for clarity.*

In the following we illustrate the typing rules, by defining side conditions, explaining the notation, and relating the rules to the semantics of choreographies in Section 3. Fig. 1 collects all the rules for convenience; in commenting the rules, we motivate their soundness (cf. Theorem 12).

$$\frac{}{\emptyset \vdash \mathbf{0} : \langle \emptyset, \emptyset \rangle}\ \text{T-EMP} \qquad \frac{\phi = \Lambda = \{\mathsf{A}\,\mathsf{B}!\mathsf{m}, \mathsf{A}\,\mathsf{B}?\mathsf{m}\}}{\{\mathsf{A},\mathsf{B}\} \vdash \mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B} : \langle \phi, \Lambda \rangle}\ \text{T-INT}$$

$$\frac{\Pi_1 \vdash \mathsf{G}_1 : \langle \phi_1, \Lambda_1 \rangle \qquad \Pi_2 \vdash \mathsf{G}_2 : \langle \phi_2, \Lambda_2 \rangle}{\Pi_1 \cup \Pi_2 \vdash \mathsf{G}_1 ; \mathsf{G}_2 : \langle \phi_1 \cup (\phi_2 - \Pi_1), \Lambda_2 \cup (\Lambda_1 - \Pi_2) \rangle}\ \text{T-SEQ}$$

$$\frac{\Pi_1 \vdash \mathsf{G}_1 : \langle \phi_1, \Lambda_1 \rangle \qquad \Pi_2 \vdash \mathsf{G}_2 : \langle \phi_2, \Lambda_2 \rangle \qquad \Pi_1 \cap \Pi_2 = \emptyset}{\Pi_1 \cup \Pi_2 \vdash \mathsf{G}_1 \mid \mathsf{G}_2 : \langle \phi_1 \cup \phi_2, \Lambda_1 \cup \Lambda_2 \rangle}\ \text{T-PAR}$$

$$\frac{\Pi \vdash \mathsf{G}_1 : \langle \phi_1, \Lambda_1 \rangle \qquad \Pi \vdash \mathsf{G}_2 : \langle \phi_2, \Lambda_2 \rangle \qquad \phi_1 \bowtie_\Pi \phi_2}{\Pi \vdash \mathsf{G}_1 + \mathsf{G}_2 : \langle \phi_1 \cup \phi_2, \Lambda_1 \cup \Lambda_2 \rangle}\ \text{T-CH}$$

Figure 1: Typing rules for g-choreographies.

Interaction Define the mapping $\widehat{\_} : 2^{\mathscr{L}} \to \mathscr{P} \to 2^{\mathscr{L}}$ by $\widehat{L}(\mathsf{A}) = \{l \in L \mid \mathsf{sbj}\ l = \mathsf{A}\}$. Then

$$L = \bigcup_{\mathsf{A} \in \mathscr{P}} \widehat{L}(\mathsf{A})$$

so that we can see any $L \subseteq \mathscr{L}$ as a family of sets of (labels of) actions indexed over $\mathscr{P}$. Note that if L is finite then $\widehat{L}(\mathsf{A}) \neq \emptyset$ for finitely many A. Now, inspecting the rule for interaction:

$$\frac{\phi = \Lambda = \{\mathsf{A}\,\mathsf{B}!\mathsf{m}, \mathsf{A}\,\mathsf{B}?\mathsf{m}\}}{\{\mathsf{A},\mathsf{B}\} \vdash \mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B} : \langle \phi, \Lambda \rangle}\ \text{T-INT}$$

we see that $\{\mathsf{A},\mathsf{B}\} = \mathscr{P}(\mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B})$; also we know that $\mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B}$ has a defined semantics (recall that $\mathscr{P}(\mathsf{G})$ are the participants occurring in G):

$$[\![\mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B}]\!] = (\{e_1, e_2\}, \{e_1 < e_2\}, \emptyset, \{e_1 \mapsto \mathsf{A}\,\mathsf{B}!\mathsf{m},\ e_2 \mapsto \mathsf{B}\,\mathsf{A}?\mathsf{m}\}) \qquad \text{hence}$$

$$\widehat{\phi}(\mathsf{A}) = \widehat{\Lambda}(\mathsf{A}) = \{\mathsf{A}\,\mathsf{B}!\mathsf{m}\} = \min([\![\mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B}]\!] \upharpoonright \mathsf{A}) \quad \text{and} \quad \widehat{\phi}(\mathsf{A}) = \widehat{\Lambda}(\mathsf{B}) = \{\mathsf{B}\,\mathsf{A}?\mathsf{m}\} = \min([\![\mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B}]\!] \upharpoonright \mathsf{B})$$

The distinction among minimal and maximal elements in a singleton poset is clearly immaterial; it becomes sensible in case of the subsequent rules.

Sequential composition If $L \subseteq \mathscr{L}$ and $\Pi \subseteq \mathscr{P}$ then set $L - \Pi = \{l \in L \mid \mathsf{sbj}\ l \notin \Pi\}$. Then the rule is:

$$\frac{\Pi_1 \vdash \mathsf{G}_1 : \langle \phi_1, \Lambda_1 \rangle \qquad \Pi_2 \vdash \mathsf{G}_2 : \langle \phi_2, \Lambda_2 \rangle}{\Pi_1 \cup \Pi_2 \vdash \mathsf{G}_1 ; \mathsf{G}_2 : \langle \phi_1 \cup (\phi_2 - \Pi_1), \Lambda_2 \cup (\Lambda_1 - \Pi_2) \rangle}\ \text{T-SEQ}$$

By induction $[\![\mathsf{G}_i]\!] = \mathscr{E}_i \neq \bot$ for $i = 1,2$, hence $[\![\mathsf{G}_1 ; \mathsf{G}_2]\!] = \mathsf{seq}([\![\mathsf{G}_1]\!], [\![\mathsf{G}_2]\!])$ is defined. Let $\mathscr{E}_i = [\![\mathsf{G}_i]\!]$ and, for each indexed over x in the set $\mathscr{C}_{\max}(\mathscr{E}_1)$ of maximal configurations of $\mathscr{E}_1$, $\mathscr{E}_x$ be disjoint event structures isomorphic to $\mathscr{E}_2$. Then by definition, the order relation $\leq_{\mathsf{seq}}$ of $\mathsf{seq}([\![\mathsf{G}_1]\!], [\![\mathsf{G}_2]\!])$ is the relation;

$$\leq_1 \ \cup \bigcup_{x \in \mathscr{C}_{\max}(\mathscr{E}_1)} \left(\leq_x \cup \{(e_1, e_2) \in x \times E_x \mid \mathsf{sbj}\ \lambda_1(e_1) = \mathsf{sbj}\ \lambda_x(e_2)\} \right)$$

where $\leq_x$ is the order relation of $\mathcal{E}_x$, and $\lambda_x(e_2) = \lambda_2(e_2)$ by construction. If e is minimal with respect to $\leq_{\mathsf{seq}}$ then either it is such with respect to $\leq_1$, or $e \in E_x$ is minimal with respect to $\leq_x$ and $\mathsf{sbj}\ \lambda_1(e') \neq \mathsf{sbj}\ \lambda_x(e)$ for all maximal $e' \in E_1$. If we consider the projection $[\![\mathsf{G}_1;\mathsf{G}_2]\!] \upharpoonright \mathsf{A}$ for any participant A of $\mathsf{G}_1;\mathsf{G}_2$, then by definition all event labels have the same subject A. So if $\mathsf{A} \in \mathscr{P}(\mathsf{G}_1)$ then each maximal configuration of $[\![\mathsf{G}_1]\!]$ has an event whose label has subject A, hence $\min([\![\mathsf{G}_1;\mathsf{G}_2]\!] \upharpoonright \mathsf{A})$ are exactly the minimal events in $\min(\mathcal{E}_1 \upharpoonright \mathsf{A})$. Otherwise, $\min([\![\mathsf{G}_1;\mathsf{G}_2]\!] \upharpoonright \mathsf{A}) = \min(\mathcal{E}_2 \upharpoonright \mathsf{A})$. By induction we know that $\Pi_i = \mathscr{P}(\mathsf{G}_i)$ and $\widehat{\phi}_i(\mathsf{A}) = \min(\mathcal{E}_i \upharpoonright \mathsf{A})$. Hence

$$\begin{array}{rcl}\min([\![\mathsf{G}_1;\mathsf{G}_2]\!] \upharpoonright \mathsf{A}) & = & \min(\mathcal{E}_1 \upharpoonright \mathsf{A}) \cup \{e \in \min(\mathcal{E}_2 \upharpoonright \mathsf{A}) \mid \mathsf{A} \notin \mathscr{P}(\mathsf{G}_1)\} \\ & = & \widehat{\phi}_1(\mathsf{A}) \cup (\widehat{\phi_2 - \Pi_1}(\mathsf{A})) \\ & = & (\phi_1 \cup \widehat{(\phi_2 - \Pi_1)})(\mathsf{A})\end{array}$$

where of course either $\widehat{\phi}_1(\mathsf{A})$ or $\widehat{\phi_2 - \Pi_1}(\mathsf{A})$ must be empty. Similarly we get $(\Lambda_2 \cup \widehat{(\Lambda_1 - \Pi_2)})(\mathsf{A}) = \max([\![\mathsf{G}_1;\mathsf{G}_2]\!] \upharpoonright \mathsf{A})$.

Example 10. Consider typing $\mathsf{C} \xrightarrow{\mathsf{req}} \mathsf{S}; \mathsf{S} \xrightarrow{\mathsf{done}} \mathsf{C}$; then we have:

$$\dfrac{\dfrac{\phi_1 = \Lambda_1 = \{\mathsf{C\,S!req}, \mathsf{C\,S?req}\}}{\{\mathsf{C},\mathsf{S}\} \vdash \mathsf{C} \xrightarrow{\mathsf{req}} \mathsf{S} : \langle \phi_1, \Lambda_1 \rangle} \qquad \dfrac{\phi_2 = \Lambda_2 = \{\mathsf{S\,C!done}, \mathsf{S\,C?done}\}}{\{\mathsf{C},\mathsf{S}\} \vdash \mathsf{S} \xrightarrow{\mathsf{done}} \mathsf{C} : \langle \phi_2, \Lambda_2 \rangle}}{\{\mathsf{C},\mathsf{S}\} \vdash \mathsf{C} \xrightarrow{\mathsf{req}} \mathsf{S}; \mathsf{S} \xrightarrow{\mathsf{done}} \mathsf{C} : \langle \phi_1, \Lambda_2 \rangle}$$

because $\{\mathsf{C},\mathsf{S}\} \cup \{\mathsf{C},\mathsf{S}\} = \{\mathsf{C},\mathsf{S}\}$, $\phi_1 \cup (\phi_2 - \{\mathsf{C},\mathsf{S}\}) = \phi_1$ since $\phi_2 - \{\mathsf{C},\mathsf{S}\} = \emptyset$, and $\Lambda_2 \cup (\Lambda_1 - \{\mathsf{C},\mathsf{S}\}) = \Lambda_2$ as $\Lambda_1 - \{\mathsf{C},\mathsf{S}\} = \emptyset$. Similarly, typing $\mathsf{C} \xrightarrow{\mathsf{md}} \mathsf{B}; \mathsf{B} \xrightarrow{\mathsf{md}} \mathsf{S}$ we obtain:

$$\dfrac{\dfrac{\phi_3 = \Lambda_3 = \{\mathsf{C\,B!md}, \mathsf{C\,B?md}\}}{\{\mathsf{C},\mathsf{B}\} \vdash \mathsf{C} \xrightarrow{\mathsf{md}} \mathsf{B} : \langle \phi_3, \Lambda_3 \rangle} \qquad \dfrac{\phi_4 = \Lambda_4 = \{\mathsf{B\,S!md}, \mathsf{B\,S?md}\}}{\{\mathsf{B},\mathsf{S}\} \vdash \mathsf{S} \xrightarrow{\mathsf{done}} \mathsf{C} : \langle \phi_4, \Lambda_4 \rangle}}{\{\mathsf{B},\mathsf{C},\mathsf{S}\} \vdash \mathsf{C} \xrightarrow{\mathsf{md}} \mathsf{B}; \mathsf{B} \xrightarrow{\mathsf{md}} \mathsf{S} : \langle \phi_5, \Lambda_5 \rangle}$$

where

$$\begin{array}{rcl}\phi_5 = \phi_3 \cup (\phi_4 - \{\mathsf{C},\mathsf{B}\}) = \phi_3 \cup \{\mathsf{B\,S?md}\} & = & \{\mathsf{C\,B!md}, \mathsf{C\,B?md}, \mathsf{B\,S?md}\} \quad \text{and} \\ \Lambda_5 = \Lambda_4 \cup (\Lambda_3 - \{\mathsf{B},\mathsf{S}\}) = \Lambda_4 \cup \{\mathsf{C\,B!md}\} & = & \{\mathsf{C\,B!md}, \mathsf{B\,S!md}, \mathsf{B\,S?md}\}\end{array}$$

Parallel composition The rule is:

$$\dfrac{\Pi_1 \vdash \mathsf{G}_1 : \langle \phi_1, \Lambda_1 \rangle \qquad \Pi_2 \vdash \mathsf{G}_2 : \langle \phi_2, \Lambda_2 \rangle \qquad \Pi_1 \cap \Pi_2 = \emptyset}{\Pi_1 \cup \Pi_2 \vdash \mathsf{G}_1 \mid \mathsf{G}_2 : \langle \phi_1 \cup \phi_2, \Lambda_1 \cup \Lambda_2 \rangle}\ \text{T-PAR}$$

By induction we may suppose that, for $i = 1,2$, Π_i equals $\mathscr{P}(\mathsf{G}_i)$ and $[\![\mathsf{G}_i]\!] = \mathcal{E}_i \neq \perp$. Hence, the condition $\Pi_1 \cap \Pi_2 = \emptyset$ implies that $\lambda_1(E_1) \cap \lambda_2(E_2) = \emptyset$, where E_i and λ_i are the carrier and the labeling mapping of $\mathcal{E}_i$ respectively, so that $[\![\mathsf{G}_1 \mid \mathsf{G}_2]\!] = [\![\mathsf{G}_1]\!] \otimes [\![\mathsf{G}_2]\!]$ is defined. Again by induction, for all participants A of G_i we have that:

$$\widehat{\phi}_i(\mathsf{A}) = \min([\![\mathsf{G}_i]\!] \upharpoonright \mathsf{A}) \qquad \text{and} \qquad \widehat{\Lambda}_i(\mathsf{A}) = \max([\![\mathsf{G}_i]\!] \upharpoonright \mathsf{A})$$

By definition of the tensor product, we know that $\leq_{\mathscr{E}_1 \otimes \mathscr{E}_2}$ is just $\leq_{\mathscr{E}_1} \cup \leq_{\mathscr{E}_2}$, which is a disjoint union (and the same holds of the $\#_{\mathscr{E}_1 \otimes \mathscr{E}_2}$ relation). Observing that

$$\widehat{\phi_1 \cup \phi_2}(\mathsf{A}) = \{l \in \phi_1 \cup \phi_2 \mid \mathsf{sbj}\ l = \mathsf{A}\} = \{l \in \phi_1 \mid \mathsf{sbj}\ l = \mathsf{A}\} \cup \{l \in \phi_2 \mid \mathsf{sbj}\ l = \mathsf{A}\} = \widehat{\phi_1}(\mathsf{A}) \cup \widehat{\phi_2}(\mathsf{A})$$

and similarly that $\widehat{\Lambda_1 \cup \Lambda_2}(\mathsf{A}) = \widehat{\Lambda_1}(\mathsf{A}) \cup \widehat{\Lambda_2}(\mathsf{A})$, we conclude that, for all participants A of $\mathsf{G}_1 \mid \mathsf{G}_2$:

$$\widehat{\phi_1 \cup \phi_2}(\mathsf{A}) = \min([\![\mathsf{G}_1 \mid \mathsf{G}_2]\!] \restriction \mathsf{A}) \qquad \text{and} \qquad \widehat{\Lambda_1 \cup \Lambda_2}(\mathsf{A}) = \max([\![\mathsf{G}_1 \mid \mathsf{G}_2]\!] \restriction \mathsf{A}).$$

Choice Two sets of labels $U, V \subseteq \mathscr{L}$ are *output uniform* if $U \cap V = \emptyset$ and $U \cup V \subseteq \mathscr{L}^!$; likewise, U and V are *input uniform* if $U \cap V = \emptyset$ and $U \cup V \subseteq \mathscr{L}^?$. Then the rule for typing choice is:

$$\frac{\Pi \vdash \mathsf{G}_1 : \langle \phi_1, \Lambda_1 \rangle \qquad \Pi \vdash \mathsf{G}_2 : \langle \phi_2, \Lambda_2 \rangle \qquad \phi_1 \bowtie_\Pi \phi_2}{\Pi \vdash \mathsf{G}_1 + \mathsf{G}_2 : \langle \phi_1 \cup \phi_2, \Lambda_1 \cup \Lambda_2 \rangle}\ \text{T-CH}$$

where the condition $\phi_1 \bowtie_\Pi \phi_2$ is defined by the clauses:

1. there is a unique $\mathsf{A} \in \Pi$ such that $\widehat{\phi_1}(\mathsf{A})$ and $\widehat{\phi_2}(\mathsf{A})$ are output uniform and both non-empty;
2. for all $\mathsf{B} \neq \mathsf{A} \in \Pi$, $\widehat{\phi_1}(\mathsf{B})$ and $\widehat{\phi_2}(\mathsf{B})$ are input uniform and $\widehat{\phi_1}(\mathsf{B}) = \emptyset$ if and only if $\widehat{\phi_2}(\mathsf{B}) = \emptyset$.

By induction, for $i = 1,2$ $[\![\mathsf{G}_i]\!] \neq \bot$, the participants of G_i are Π and $\widehat{\phi_i}(\mathsf{R}) = \min(\mathscr{E}_i \restriction \mathsf{R})$ and $\widehat{\Lambda_i}(\mathsf{R}) = \max(\mathscr{E}_i \restriction \mathsf{R})$ for all $\mathsf{R} \in \Pi$. Let $[\![\mathsf{G}_i]\!] = \mathscr{E}_i = (E_i, \leq_i, \#_i, \lambda_i)$ and $\mathscr{E} = \mathscr{E}_0 + \mathscr{E}_1$. By condition 1 above, and remembering the identification of e with $\lambda_i(e)$, we have that $\widehat{\phi_i}(\mathsf{A}) = \min(\mathscr{E}_i \restriction \mathsf{A}) \neq \emptyset$ for both $i = 1,2$, so that $\mathscr{E}_0, \mathscr{E}_1 \neq \varepsilon$.

Again by 1, we know that A is active in both $\mathscr{E}_i$, since $\min(\mathscr{E}_1 \restriction \mathsf{A})$ and $\min(\mathscr{E}_2 \restriction \mathsf{A})$ are output uniform, while condition 2 implies that all $\mathsf{B} \in \Pi \setminus \mathsf{A}$ are passive, since $\min(\mathscr{E}_i \restriction \mathsf{B})$ are input uniform. We conclude that $wb(\mathscr{E}_0, \mathscr{E}_1)$ and hence that $[\![\mathsf{G}_1 + \mathsf{G}_2]\!] \neq \bot$. Now that $\widehat{\phi_1 \cup \phi_2}(\mathsf{R}) = \min([\![\mathsf{G}_1 + \mathsf{G}_2]\!] \restriction \mathsf{R})$ and $\widehat{\Lambda_1 \cup \Lambda_2}(\mathsf{R}) = \max([\![\mathsf{G}_1 + \mathsf{G}_2]\!] \restriction \mathsf{R})$ for all $\mathsf{R} \in \mathscr{P}(\mathsf{G}_1 + \mathsf{G}_2)$ follows by induction.

Example 11. By rule T-CH we can type e.g.

$$\frac{\dfrac{\phi_1 = \Lambda_1 = \{\mathsf{C}\,\mathsf{S}!\mathsf{req}, \mathsf{C}\,\mathsf{S}?\mathsf{req}\}}{\{\mathsf{C},\mathsf{S}\} \vdash \mathsf{C} \xrightarrow{\mathsf{req}} \mathsf{S} : \langle \phi_1, \Lambda_1 \rangle} \qquad \dfrac{\phi_6 = \Lambda_6 = \{\mathsf{C}\,\mathsf{S}!\mathsf{done}, \mathsf{C}\,\mathsf{S}?\mathsf{done}\}}{\{\mathsf{C},\mathsf{S}\} \vdash \mathsf{C} \xrightarrow{\mathsf{done}} \mathsf{S} : \langle \phi_6, \Lambda_6 \rangle} \qquad \phi_1 \bowtie_{\{\mathsf{C},\mathsf{S}\}} \phi_6}{\{\mathsf{C},\mathsf{S}\} \vdash \mathsf{C} \xrightarrow{\mathsf{req}} \mathsf{S} + \mathsf{C} \xrightarrow{\mathsf{done}} \mathsf{S} : \langle \phi_1 \cup \phi_2, \Lambda_1 \cup \Lambda_2 \rangle}$$

in fact, $\phi_1 \bowtie_{\{\mathsf{C},\mathsf{S}\}} \phi_6$ holds, being C the unique participant in $\{\mathsf{C},\mathsf{S}\}$ such that $\widehat{\phi_1}(\mathsf{C})$ and $\widehat{\phi_6}(\mathsf{C})$ are output uniform, and the remaining S is such that $\widehat{\phi_1}(\mathsf{S})$ and $\widehat{\phi_6}(\mathsf{S})$ are input uniform. However none of the following are typable; recall that $\{\mathsf{C},\mathsf{S}\} \vdash \mathsf{C} \xrightarrow{\mathsf{req}} \mathsf{S} : \langle \phi_1, \Lambda_1 \rangle$, $\{\mathsf{C},\mathsf{S}\} \vdash \mathsf{S} \xrightarrow{\mathsf{done}} \mathsf{C} : \langle \phi_2, \Lambda_2 \rangle$ and $\{\mathsf{C},\mathsf{B}\} \vdash \mathsf{C} \xrightarrow{\mathsf{md}} \mathsf{B} : \langle \phi_3, \Lambda_3 \rangle$. Then:

$\mathsf{G}_1 \equiv \mathsf{C} \xrightarrow{\mathsf{req}} \mathsf{S} + \mathsf{C} \xrightarrow{\mathsf{req}} \mathsf{S}$: this is because $\widehat{\phi_1}(\mathsf{C})$ cannot be disjoint from itself;

$\mathsf{G}_2 \equiv \mathsf{C} \xrightarrow{\mathsf{req}} \mathsf{S} + \mathsf{S} \xrightarrow{\mathsf{done}} \mathsf{C}$: in this case we have that neither $\widehat{\phi_1}(\mathsf{C}) \cup \widehat{\phi_2}(\mathsf{C}) = \{\mathsf{C}\,\mathsf{S}!\mathsf{req}, \mathsf{S}\,\mathsf{C}?\mathsf{done}\}$ nor $\widehat{\phi_1}(\mathsf{S}) \cup \widehat{\phi_2}(\mathsf{S}) = \{\mathsf{C}\,\mathsf{S}?\mathsf{req}, \mathsf{S}\,\mathsf{C}!\mathsf{done}\}$ are output uniform;

$\mathsf{G}_3 \equiv \mathsf{C} \xrightarrow{\mathsf{req}} \mathsf{S} + \mathsf{C} \xrightarrow{\mathsf{md}} \mathsf{B}$: because $\{\mathsf{C},\mathsf{S}\} \neq \{\mathsf{C},\mathsf{B}\}$.

A more complex case is the following (continuing example 10):

$$\dfrac{\dfrac{\phi_1 = \Lambda_1 = \{\mathsf{C}\,\mathsf{S}!\mathsf{md}, \mathsf{C}\,\mathsf{S}?\mathsf{md}\}}{\{\mathsf{C},\mathsf{S}\} \vdash \mathsf{C} \xrightarrow{\mathsf{md}} \mathsf{S} : \langle \phi_1, \Lambda_1 \rangle} \qquad \{\mathsf{C},\mathsf{S}\} \vdash \mathsf{C} \xrightarrow{\mathsf{req}} \mathsf{S}; \mathsf{S} \xrightarrow{\mathsf{done}} \mathsf{C} : \langle \phi_2, \Lambda_3 \rangle}{\{\mathsf{C},\mathsf{S}\} \vdash \mathsf{C} \xrightarrow{\mathsf{md}} \mathsf{S} + \mathsf{C} \xrightarrow{\mathsf{req}} \mathsf{S}; \mathsf{S} \xrightarrow{\mathsf{done}} \mathsf{C} : \langle \phi_1 \cup \phi_2, \Lambda_1 \cup \Lambda_3 \rangle}$$

because C is the unique participant in $\{\mathsf{C},\mathsf{S}\}$ such that $\widehat{\phi_1}(\mathsf{C})$ and $\widehat{\phi_2}(\mathsf{C})$ are output uniform, and the remaining S is such that $\widehat{\phi_1}(\mathsf{S})$ and $\widehat{\phi_2}(\mathsf{S})$ are input uniform, namely condition $\phi_1 \bowtie_{\{\mathsf{C},\mathsf{S}\}} \phi_2$ is satisfied.

On the other hand the choreography $\mathsf{C} \xrightarrow{\mathsf{md}} \mathsf{B}; \mathsf{B} \xrightarrow{\mathsf{md}} \mathsf{S} + \mathsf{C} \xrightarrow{\mathsf{req}} \mathsf{S}; \mathsf{S} \xrightarrow{\mathsf{done}} \mathsf{C}$ is not typeable because knowing from example 10 that

$$\{\mathsf{B},\mathsf{C},\mathsf{S}\} \vdash \mathsf{C} \xrightarrow{\mathsf{md}} \mathsf{B}; \mathsf{B} \xrightarrow{\mathsf{md}} \mathsf{S} : \langle \phi_6, \Lambda_6 \rangle$$

we have that $\{\mathsf{B},\mathsf{C},\mathsf{S}\} \neq \{\mathsf{C},\mathsf{S}\}$, so that rule T-CH doesn't apply.

In summary we have proved the following result.

Theorem 12 (Soundness)**.** *If* $\Pi \vdash \mathsf{G} : \langle \phi, \Lambda \rangle$ *is derivable then* $[\![\mathsf{G}]\!] \neq \bot$, $\Pi = \mathscr{P}(\mathsf{G})$, *and*

$$\widehat{\phi}(\mathsf{A}) = \min([\![\mathsf{G}]\!] \restriction \mathsf{A}) \qquad \textit{and} \qquad \widehat{\Lambda}(\mathsf{A}) = \max([\![\mathsf{G}]\!] \restriction \mathsf{A})$$

holds for all $\mathsf{A} \in \Pi$.

Corollary 13. *If* G *is a typable g-choreography then* G *is well-formed. Moreover a typable* G *has exactly one typing* $\Pi \vdash \mathsf{G} : \langle \phi, \Lambda \rangle$.

Our typing system is not complete. For instance, the g-choreography $\mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B} \mid \mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{C}$ is well-formed but it cannot be typed because the rule T-PAR cannot be applied since A occurs on both sides of the parallel. In fact, this is the only obstacle to attain completeness and could be removed by tracing in the types not only the minimal and maximal communications, but also the communications of threads.

Lemma 14. *If* $\Pi \vdash \mathsf{G} : \langle \phi, \Lambda \rangle$ *is derivable then, for all* $x \in \mathscr{C}_{\max}([\![\mathsf{G}]\!])$ *and* $\mathsf{A} \in \mathscr{P}$, *either* $\max(x \restriction \mathsf{A})$ *is empty or it is a singleton.*

Proof. By induction on the derivation of $\Pi \vdash \mathsf{G} : \langle \phi, \Lambda \rangle$ and case analysis of the last typing rule applied observing that rule T-PAR requires to partition the context in two disjoint sub-contexts. □

We remark that such a completeness result would be basically due to the strictness of the conditions of Definition 4. In fact, more general notions of well-branchedness would break the completeness theorem. For instance, we can weaken the conditions of Definition 4 as follows.

- The projections on the event structures of the two branches may either be disjoint inputs (as per the current determined choice condition) or be isomorphic
- there is a unique selector (as currently required in Definition 4) and any other participant whose minimal actions are output have isomorphic projections on the two branches.

With this change, the g-choreography $\mathsf{G} = \mathsf{G}_1 + \mathsf{G}_2$ where $\mathsf{G}_1 = \mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B}; \mathsf{C} \xrightarrow{\mathsf{x}} \mathsf{B}$ and $\mathsf{G}_2 = \mathsf{A} \xrightarrow{\mathsf{n}} \mathsf{B}; \mathsf{C} \xrightarrow{\mathsf{x}} \mathsf{B}$ would become well formed. However, our system cannot type G since both A and C are selectors in the choice of G. Notice that G_1 violates the unique selector condition of Definition 4, while it does not violate the more general conditions above since C behaves the same in G_1 and G_2.

4 Refinement

To the grammar of Definition 3 we add a new construct that we dub *refinable action*:

$$\mathsf{G} \ ::= \ \cdots \mid \mathsf{A} \xrightarrow{\mathsf{m_1 \ldots m_n}} \mathsf{B_1 \ldots B_n} \qquad\qquad \text{refinable action}$$

where $\overline{\mathsf{m}} = \mathsf{m}_1, \ldots, \mathsf{m}_n$ and $\overline{\mathsf{B}} = \mathsf{B}_1, \ldots, \mathsf{B}_n$ are non-empty tuples of the same lenght of messages and participants such that the participants in $\overline{\mathsf{B}}$ are pairwise distinct. Call *refinable* a g-choreography generated by the so extended grammar; a g-choreography is *ground* or *non-refinable* if it is derivable only with the productions of the grammar in Definition 3.

Definition 8 (Refines relation)**.** *A ground g-choreography* G refines $\mathsf{A} \xrightarrow{\overline{\mathsf{m}}} \overline{\mathsf{B}}$*, written* G ref $\mathsf{A} \xrightarrow{\overline{\mathsf{m}}} \overline{\mathsf{B}}$*, if*

1. $[\![\mathsf{G}]\!] = \mathscr{E} \neq \bot$*;*
2. $\mathit{sbj}\,\min(\mathscr{E}) = \{\mathsf{A}\}$*, by which we say that* A *is the (unique)* initiator *of* G*;*
3. *letting* $\overline{\mathsf{m}} = \mathsf{m}_1, \ldots, \mathsf{m}_n$ *and* $\overline{\mathsf{B}} = \mathsf{B}_1, \ldots, \mathsf{B}_n$*, for all* $x \in \mathscr{C}_{\max}(\mathscr{E})$ *and* $1 \leq h \leq n$ *there exists* $\mathsf{C} \in \mathscr{P}(\mathsf{G})$ *such that* $\mathsf{C}\,\mathsf{B_h}?\mathsf{m_h} \in \max(x \restriction \mathsf{B}_h)$.

In words, G refines $\mathsf{A} \xrightarrow{\overline{\mathsf{m}}} \overline{\mathsf{B}}$ if G is meaningful, that is well-formed, with a unique participant A initiating the interaction by some (necessarily distinct) output actions, and such that in all branches, namely maximal configurations x of $[\![\mathsf{G}]\!]$, each B_h eventually inputs m_h.

Example 15. The following

$$\mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B} \text{ ref } \mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B}, \quad \mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B} + \mathsf{A} \xrightarrow{\mathsf{n}} \mathsf{B}; \mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B} \text{ ref } \mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B}, \quad \text{and} \quad \mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B} \mid \mathsf{C} \xrightarrow{\mathsf{n}} \mathsf{B} \text{ ref } \mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B}$$

are examples of refinement relations. ⋄

Our next step is to devise sufficient conditions for substituting the refinement action $\mathsf{A} \xrightarrow{\overline{\mathsf{m}}} \overline{\mathsf{B}}$ by some of its ground refinements G' in a context of the shape $\mathsf{G}[\mathsf{A} \xrightarrow{\overline{\mathsf{m}}} \overline{\mathsf{B}}]$ while ensuring well-formednes of the resulting g-choreography $\mathsf{G}[\mathsf{G}']$. In view of the previous section, an eligible tool is the typing system. Hereafter, let $\mathsf{G}'[\cdot]$ be a ground g-choreography with a hole $[\cdot]$, namely a placeholder such that, if replaced by a ground G the resulting $\mathsf{G}'[\mathsf{G}]$ is a ground g-chreography.

In fact, observe that, barred for the axioms, the shape of the typing judgement $\Pi \vdash \mathsf{G} : \langle \phi, \Lambda \rangle$ in the conclusion of each rule only depends on contexts and types in the premises. Combining this remark with Theorem 12, we get the following corollary.

Corollary 16. *Suppose that* $\Pi' \vdash \mathsf{G}'[\cdot] : \langle \phi', \Lambda' \rangle$ *is derivable in the type system extended by the axiom* $\Pi \vdash [\cdot] : \langle \phi, \Lambda \rangle$*. Then for all* G *such that* $\Pi \vdash \mathsf{G} : \langle \phi, \Lambda \rangle$ *is derivable, the judgment* $\Pi' \vdash \mathsf{G}'[\mathsf{G}] : \langle \phi', \Lambda' \rangle$ *is derivable, so that* $[\![\mathsf{G}'[\mathsf{G}]]\!] \neq \bot$.

To put this corollary to use, we have to define an axiom schema for deducing $\Pi \vdash \mathsf{A} \xrightarrow{\overline{\mathsf{m}}} \overline{\mathsf{B}} : \langle \phi, \Lambda \rangle$ such that if $\Pi \vdash \mathsf{G} : \langle \phi, \Lambda \rangle$ is derivable for some ground G, then G refines $\mathsf{A} \xrightarrow{\overline{\mathsf{m}}} \overline{\mathsf{B}}$. To do that we need a preliminary lemma.

Lemma 17. *Let* $[\![\mathsf{G}]\!] \neq \varepsilon$ *be ground and well-formed. If* $x \in \mathscr{C}_{\max}([\![\mathsf{G}]\!])$ *then*

1. $\emptyset \neq \min(x) \subseteq \mathscr{L}^{!}$ *and* $\emptyset \neq \max(x) \subseteq \mathscr{L}^{?}$*;*
2. $\mathit{sbj}\,x = \mathscr{P}(\mathsf{G})$.

Proof. By induction over G.

If $\mathsf{G} \equiv \mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B}$ then $[\![\mathsf{G}]\!] = \{\mathsf{A\,B!m} < \mathsf{A\,B?m}\}$ and the (1)-(2) are immediately verified.

If $\mathsf{G} \equiv \mathsf{G}_1 \mid \mathsf{G}_2$ we have that $[\![\mathsf{G}]\!] = [\![\mathsf{G}_1]\!] \otimes [\![\mathsf{G}_2]\!]$ by well-formednes. If either of the $[\![\mathsf{G}_i]\!]$ is ε the thesis follows immediately by induction, since the other one has to be $\neq \varepsilon$. Suppose that $[\![\mathsf{G}_i]\!] \neq \varepsilon$ for both $i = 1,2$. By definition of $\otimes$ we have that for some non empty $x_i \in \mathscr{C}_{\max}([\![\mathsf{G}_i]\!])$ it is $x = x_1 \cup x_2$, from which part (1) of the lemma follows by induction. Similarly if $y \in \mathscr{C}_{\max}([\![\mathsf{G}]\!])$ then $y = y_1 \cup y_2$ for $y_i \in \mathscr{C}_{\max}([\![\mathsf{G}_i]\!])$, hence by induction $\mathsf{sbj}\ x_i = \mathsf{sbj}\ y_i$ for $i = 1,2$, so that $\mathsf{sbj}\ x = \mathsf{sbj}\ x_1 \cup \mathsf{sbj}\ x_2 = \mathsf{sbj}\ y_1 \cup \mathsf{sbj}\ y_2 = \mathsf{sbj}\ y$ and we conclude that (2) holds.

If $\mathsf{G} \equiv \mathsf{G}_1;\mathsf{G}_2$ then by definition of $[\![\mathsf{G}]\!] = \mathsf{seq}([\![\mathsf{G}_1]\!],[\![\mathsf{G}_2]\!])$ any $x \in \mathscr{C}_{\max}([\![\mathsf{G}]\!])$ either $x \in \mathscr{C}_{\max}([\![\mathsf{G}_1]\!])$, or $x \in \mathscr{C}_{\max}([\![\mathsf{G}_2]\!])$ or there exist $x_1 \in \mathscr{C}_{\max}([\![\mathsf{G}_1]\!])$ and $x_2 \in \mathscr{C}_{\max}([\![\mathsf{G}_2]\!])$ with $x = x_1 \cup x_2$ and $\min(x) = \min(x_1)$, $\max(x) = \max(x_2)$. Now (1) follows by induction. To see (2) suppose that $y \in \mathscr{C}_{\max}([\![\mathsf{G}]\!])$ and, toward a contradiction, assume that $\mathsf{sbj}\ x \neq \mathsf{sbj}\ y$: this is only possible if say $x \in \mathscr{C}_{\max}([\![\mathsf{G}_1]\!]) \setminus \mathscr{C}_{\max}([\![\mathsf{G}_2]\!])$ and $y \in \mathscr{C}_{\max}([\![\mathsf{G}_2]\!]) \setminus \mathscr{C}_{\max}([\![\mathsf{G}_1]\!])$, since otherwise if e.g. $x = x_1 \cup x_2$ as above then $\mathsf{sbj}\ y = \mathsf{sbj}\ x_2$ by induction, but then by definition of $\mathsf{seq}([\![\mathsf{G}_1]\!],[\![\mathsf{G}_2]\!])$ there is a pair of events $e_1 \in x_1$ and $e_2 \in x_2$ with $\mathsf{sbj}\ \lambda_1(e_1) = \mathsf{sbj}\ \lambda_2(e_2)$ so that by construction $e_1 \leq_{[\![\mathsf{G}]\!]} e_2$. This implies that there exist $e_3 \in y$ s.t. $\mathsf{sbj}\ \lambda_2(e_3) = \mathsf{sbj}\ \lambda_2(e_2)$ so that $e_1 \leq_{[\![\mathsf{G}]\!]} e_3$ contradicting the maximality of $y \in \mathscr{C}_{\max}([\![\mathsf{G}]\!])$.

But if $x \in \mathscr{C}_{\max}([\![\mathsf{G}_1]\!]) \setminus \mathscr{C}_{\max}([\![\mathsf{G}_2]\!])$ and $y \in \mathscr{C}_{\max}([\![\mathsf{G}_2]\!]) \setminus \mathscr{C}_{\max}([\![\mathsf{G}_1]\!])$ then $x \neq y$ and $x \cup y \in \mathscr{C}_{\max}([\![\mathsf{G}]\!])$ contradicting the hypothesis that $x, y \in \mathscr{C}_{\max}([\![\mathsf{G}]\!])$.

If $\mathsf{G} \equiv \mathsf{G}_1 + \mathsf{G}_2$ and it is well-formed then $\mathscr{P}(\mathsf{G}_1) = \mathscr{P}(\mathsf{G}_2) = \mathscr{P}(\mathsf{G})$ and there exists a unique active $\mathsf{A} \in \mathscr{P}(\mathsf{G})$ in $[\![\mathsf{G}]\!] = [\![\mathsf{G}_1]\!] + [\![\mathsf{G}_2]\!]$ that is the subject of all events in $\min(x)$ for any $x \in \mathscr{C}_{\max}([\![\mathsf{G}]\!])$. This also implies that $[\![\mathsf{G}]\!] \neq \varepsilon$, so that at least one of the two $[\![\mathsf{G}_i]\!]$ is such. By induction we have immediately (1). To prove (2) let $y \in \mathscr{C}_{\max}([\![\mathsf{G}]\!])$. By definition of $[\![\mathsf{G}_1]\!] + [\![\mathsf{G}_2]\!]$ we have that $\mathscr{C}_{\max}([\![\mathsf{G}]\!]) = \mathscr{C}_{\max}([\![\mathsf{G}_1]\!]) \cup \mathscr{C}_{\max}([\![\mathsf{G}_2]\!])$, so that either x and y belong to the same $\mathscr{C}_{\max}([\![\mathsf{G}_i]\!])$, then $\mathsf{sbj}\ x = \mathsf{sbj}\ y$ by induction, or say $x \in \mathscr{C}_{\max}([\![\mathsf{G}_1]\!])$ and $y \in \mathscr{C}_{\max}([\![\mathsf{G}_2]\!])$: then by induction $\mathsf{sbj}\ x = \mathscr{P}(\mathsf{G}_1) = \mathscr{P}(\mathsf{G}_2) = \mathsf{sbj}\ y$ and we are done. □

Lemma 18. *Let* $\mathsf{A} \overset{\overline{\mathsf{m}}}{\dashrightarrow} \overline{\mathsf{B}}$ *be a refinable action. If* $\Pi \subseteq \mathscr{P}$ *and* $\phi, \Lambda \subseteq \mathscr{L}$ *are such that*

1. $\mathsf{sbj}\ \phi = \mathsf{sbj}\ \Lambda = \Pi$,
2. $\mathsf{sbj}\ (\phi \cap \mathscr{L}^!) = \{\mathsf{A}\}$, *and*
3. *assuming* $\overline{\mathsf{B}} = \mathsf{B}_1, \ldots, \mathsf{B}_n$ *and* $\overline{\mathsf{m}} = \mathsf{m}_1, \ldots, \mathsf{m}_n$, *for all* $1 \leq h \leq n$ *there exists* C *such that* $\widehat{\Lambda}(\mathsf{B_h}) = \{\mathsf{C\,B_h?m_h}\}$

then $\Pi \vdash \mathsf{G} : \langle \phi, \Lambda \rangle$ *implies* $\mathsf{G}\ \mathsf{ref}\ \mathsf{A} \overset{\overline{\mathsf{m}}}{\dashrightarrow} \overline{\mathsf{B}}$.

Proof. By Theorem 12, we know that $\Pi \vdash \mathsf{G} : \langle \phi, \Lambda \rangle$ implies $[\![\mathsf{G}]\!] \neq \bot$; on the other hand the hypotheses imply that none among Π, ϕ and Λ is empty, hence $[\![\mathsf{G}]\!] \neq \varepsilon$. Recall that, by the same theorem, $\widehat{\phi}(\mathsf{C}) = \min([\![\mathsf{G}]\!] \restriction \mathsf{C})$ and $\widehat{\Lambda}(\mathsf{C}) = \max([\![\mathsf{G}]\!] \restriction \mathsf{C})$ for all $\mathsf{C} \in \Pi = \mathscr{P}(\mathsf{G})$.

Let $x \in \mathscr{C}_{\max}([\![\mathsf{G}]\!])$. We have $\emptyset \neq \min(x) \subseteq \phi \cap \mathscr{L}^!$ by the above and by Lemma 17.1, hence $\mathsf{sbj}\ (\phi \cap \mathscr{L}^!) = \{\mathsf{A}\}$ implies $\mathsf{sbj}\ \min(x) = \{\mathsf{A}\}$, and therefore A is the initiator of G.

On the other hand, by the hypothesis that for all $\mathsf{B_h} \in \overline{\mathsf{B}}$ there exists C s.t. $\widehat{\Lambda}(\mathsf{B_h}) = \{\mathsf{C\,B_h?m_h}\}$ an Theorem 12 we infer that $\{\mathsf{C\,B_h?m_h}\} = \widehat{\Lambda}(\mathsf{B_h}) = \max([\![\mathsf{G}]\!] \restriction \mathsf{B_h})$ for all $\mathsf{B_h} \in \overline{\mathsf{B}}$. Therefore there exists $y \in \mathscr{C}_{\max}([\![\mathsf{G}]\!])$ such that $\mathsf{C\,B_h?m_h} \in y \cap \Lambda$; by Lemma 17.2, $\mathsf{sbj}\ x = \mathsf{sbj}\ y$, hence $\mathsf{B_h} \in \mathsf{sbj}\ x$ and so $\widehat{x}(\mathsf{B_h}) \neq \emptyset$. This and the the typability of G imply that $\widehat{x}(\mathsf{B_h})$ is a singleton by Lemma 14: from this and the fact that $\max(x \restriction \mathsf{B_h}) \subseteq \Lambda$ it follows that $\max(x \restriction \mathsf{B_h}) = \widehat{\Lambda}(\mathsf{B_h}) = \{\mathsf{C\,B_h?m_h}\}$. □

In view of Lemma 18 it is sound to extend the type system to refinable choreographies by adding to the rules in Figure 1 the following axiom schema:

$$\frac{\mathsf{sbj}\ \phi = \mathsf{sbj}\ \Lambda = \Pi \quad \mathsf{sbj}\ (\phi \cap \mathscr{L}^!) = \{\mathsf{A}\} \quad \forall \mathsf{h} \exists \mathsf{C} \in \Pi.\ \widehat{\Lambda}(\mathsf{B_h}) = \{\mathsf{C}\,\mathsf{B_h}?\mathsf{m_h}\}}{\Pi \vdash \mathsf{A} \xrightarrow{\mathsf{m_1}\ldots\mathsf{m_n}} \mathsf{B_1}\ldots\mathsf{B_n} : \langle \phi, \Lambda \rangle}\ \text{T-REF}$$

Remark 19. *Given any refinable action $\mathsf{A} \xrightarrow{\overline{\mathsf{m}}} \overline{\mathsf{B}} \equiv \mathsf{A} \xrightarrow{\mathsf{m_1}\ldots\mathsf{m_n}} \mathsf{B_1}\ldots\mathsf{B_n}$ there is a typing context Π_0, ϕ_0, Λ_0 such that $\Pi_0 \vdash \mathsf{A} \xrightarrow{\overline{\mathsf{m}}} \overline{\mathsf{B}} : \langle \phi_0, \Lambda_0 \rangle$ is an instance of rule* T-REF*. Indeed taking $\Pi_0 = \{\mathsf{A}, \mathsf{B_1}, \ldots, \mathsf{B_n}\}$ and $\phi_0 = \Lambda_0 = \{\mathsf{A}\,\mathsf{B_1}!\mathsf{m_1}, \mathsf{A}\,\mathsf{B_1}?\mathsf{m_1}, \ldots, \mathsf{A}\,\mathsf{B_n}!\mathsf{m_n}, \mathsf{A}\,\mathsf{B_n}?\mathsf{m_n}\}$ it is easy to check that the conditions of rule* T-REF *are satisfied. On the contrary the same conditions do not ensure that there is a* G *that is typable in the given context: let $\Pi' = \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}\}$ and $\phi' = \Lambda' = \{\mathsf{A}\,\mathsf{B}!\mathsf{m}, \mathsf{A}\,\mathsf{B}?\mathsf{m}, \mathsf{C}\,\mathsf{D}?\mathsf{m}\}$, then $\Pi' \vdash \mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B} : \langle \phi', \Lambda' \rangle$, but there exists no ground* G *such that $\Pi' \vdash \mathsf{G} : \langle \phi', \Lambda' \rangle$.*

Example 20. The refinements in the above proof are the simplest, but less interesting ones; to see more significant examples let us first generalize Corollary 16 to the case of contexts $\mathsf{G}[\cdot]_1 \ldots [\cdot]_n$, with n distinct holes. Suppose that $\Pi_i \vdash \mathsf{A_i} \xrightarrow{\overline{\mathsf{m_i}}} \overline{\mathsf{B_i}} : \langle \phi_i, \Lambda_i \rangle$ are the instances of T-REF that have been used in deriving:

$$\Pi \vdash \mathsf{G}[\mathsf{A_1} \xrightarrow{\overline{\mathsf{m_1}}} \overline{\mathsf{B_1}}]_1 \ldots [\mathsf{A_n} \xrightarrow{\overline{\mathsf{m_n}}} \overline{\mathsf{B_n}}]_n : \langle \phi, \Lambda \rangle$$

If $\Pi_i \vdash \mathsf{G_i} : \langle \phi_i, \Lambda_i \rangle$ are derivable for ground G_i then $\Pi \vdash \mathsf{G}[\mathsf{G_1}]_1 \ldots [\mathsf{G_n}]_n : \langle \phi, \Lambda \rangle$ is derivable, and hence $\mathsf{G}[\mathsf{G_1}]_1 \ldots [\mathsf{G}_n]_n$ is well-formed by Theorem 12.

Resuming from the Introduction and adapting from Example 11 we have:

$$\frac{\dfrac{}{\Pi \vdash \mathsf{C} \xrightarrow{\mathsf{md}} \mathsf{S} : \langle \phi_1, \Lambda_1 \rangle}\text{T-REF} \qquad \dfrac{\dfrac{}{\Pi \vdash \mathsf{C} \xrightarrow{\mathsf{req}} \mathsf{S} : \langle \phi_2, \Lambda_2 \rangle}\text{T-REF} \quad \dfrac{}{\Pi \vdash \mathsf{S} \xrightarrow{\mathsf{done}} \mathsf{C} : \langle \phi_3, \Lambda_3 \rangle}\text{T-REF}}{\Pi \vdash \mathsf{C} \xrightarrow{\mathsf{req}} \mathsf{S}; \mathsf{S} \xrightarrow{\mathsf{done}} \mathsf{C} : \langle \phi_2, \Lambda_3 \rangle}}{\Pi \vdash \mathsf{C} \xrightarrow{\mathsf{md}} \mathsf{S} + \mathsf{C} \xrightarrow{\mathsf{req}} \mathsf{S}; \mathsf{S} \xrightarrow{\mathsf{done}} \mathsf{C} : \langle \phi_1 \cup \phi_2, \Lambda_1 \cup \Lambda_3 \rangle}\text{T-CH} \tag{11}$$

where $\Pi = \{\mathsf{C}, \mathsf{S}\}$, $\phi_1 = \Lambda_1 = \{\mathsf{C}\,\mathsf{S}!\mathsf{md}, \mathsf{C}\,\mathsf{S}?\mathsf{md}\}$, $\phi_2 = \Lambda_2 = \{\mathsf{C}\,\mathsf{S}!\mathsf{req}, \mathsf{C}\,\mathsf{S}?\mathsf{req}\}$ and $\phi_3 = \Lambda_3 = \{\mathsf{S}\,\mathsf{C}!\mathsf{done}, \mathsf{S}\,\mathsf{C}?\mathsf{done}\}$. From Example 11 we know that we can derive $\Pi \vdash \mathsf{C} \xrightarrow{\mathsf{md}} \mathsf{S} : \langle \phi_1, \Lambda_1 \rangle$, $\Pi \vdash \mathsf{C} \xrightarrow{\mathsf{req}} \mathsf{S} : \langle \phi_2, \Lambda_2 \rangle$ and $\Pi : \langle \phi_3, \Lambda_3 \rangle$; hence we conclude that $\Pi \vdash \mathsf{C} \xrightarrow{\mathsf{md}} \mathsf{S} + \mathsf{C} \xrightarrow{\mathsf{req}} \mathsf{S}; \mathsf{S} \xrightarrow{\mathsf{done}} \mathsf{C} : \langle \phi_1 \cup \phi_2, \Lambda_1 \cup \Lambda_3 \rangle$ is derivable, and so semantically well-formed.

Consider now the g-choreography $\mathsf{C} \xrightarrow{\mathsf{md}} \mathsf{B}; \mathsf{B} \xrightarrow{\mathsf{md}} \mathsf{S}$, which can be checked to refine $\mathsf{C} \xrightarrow{\mathsf{md}} \mathsf{S}$: refining the latter with the former in the context $\mathsf{C} \xrightarrow{\mathsf{md}} \mathsf{S} + \mathsf{C} \xrightarrow{\mathsf{req}} \mathsf{S}; \mathsf{S} \xrightarrow{\mathsf{done}} \mathsf{C}$ spoils well-formed, and indeed $\Pi \not\vdash \mathsf{C} \xrightarrow{\mathsf{md}} \mathsf{B}; \mathsf{B} \xrightarrow{\mathsf{md}} \mathsf{S} : \langle \phi_1, \Lambda_1 \rangle$. However it suffices to take $\Pi' = \Pi \cup \{\mathsf{B}\} = \{\mathsf{C}, \mathsf{S}, \mathsf{B}\}$ and $\phi_1' = \phi_1 \cup \{\mathsf{C}\,\mathsf{B}?\mathsf{md}\}$, $\Lambda_1' = \Lambda_1 \cup \{\mathsf{B}\,\mathsf{S}!\mathsf{md}\}$ to have that $\Pi' \vdash \mathsf{C} \xrightarrow{\mathsf{md}} \mathsf{B}; \mathsf{B} \xrightarrow{\mathsf{md}} \mathsf{S} : \langle \phi_1', \Lambda_1' \rangle$ is derivable and $\Pi' \vdash \mathsf{C} \xrightarrow{\mathsf{md}} \mathsf{S} : \langle \phi_1', \Lambda_1' \rangle$ is an instance of T-REF. Incidentally we note that $\Pi', \phi_1', \Lambda_1'$ are computable from $\mathsf{C} \xrightarrow{\mathsf{md}} \mathsf{B}; \mathsf{B} \xrightarrow{\mathsf{md}} \mathsf{S}$.

Still we cannot freely replace $\Pi' \vdash \mathsf{C} \xrightarrow{\mathsf{md}} \mathsf{S} : \langle \phi_1', \Lambda_1' \rangle$ in the derivation (11), because $\Pi' \neq \Pi$ where the difference is B, making rule T-CH inapplicable. However, by computing the typing context of $\mathsf{C} \xrightarrow{\mathsf{x}} \mathsf{B}; \mathsf{B} \xrightarrow{\mathsf{req}} \mathsf{S}$ we obtain $\Pi' \vdash \mathsf{C} \xrightarrow{\mathsf{x}} \mathsf{B}; \mathsf{B} \xrightarrow{\mathsf{req}} \mathsf{S} : \langle \phi_2', \Lambda_2' \rangle$ where $\phi_2' = \{\mathsf{C}\,\mathsf{B}!\mathsf{x}, \mathsf{C}\,\mathsf{B}?\mathsf{x}, \mathsf{B}\,\mathsf{S}!\mathsf{rep}\}$ and $\Lambda_2' = \{\mathsf{C}\,\mathsf{B}?\mathsf{x}, \mathsf{B}\,\mathsf{S}!\mathsf{rep}, \mathsf{B}\,\mathsf{S}?\mathsf{rep}\}$ we can easily check that $\Pi' \vdash \mathsf{C} \xrightarrow{\mathsf{req}} \mathsf{S} : \langle \phi_2', \Lambda_2' \rangle$ is an instance of T-REF, so that $\mathsf{C} \xrightarrow{\mathsf{x}} \mathsf{B}; \mathsf{B} \xrightarrow{\mathsf{req}} \mathsf{S}$ ref $\mathsf{C} \xrightarrow{\mathsf{req}} \mathsf{S}$ by Lemma 18. In this case we do not need to further modify the derivation

(11) to deduce $\Pi \vdash \mathsf{C} \overset{\mathsf{req}}{\dashrightarrow} \mathsf{S}; \mathsf{S} \overset{\mathsf{done}}{\dashrightarrow} \mathsf{C} : \langle \phi_2', \Lambda_3' \rangle$ with $\Lambda_3' = \Lambda_3 \cup \{\mathsf{B}\,\mathsf{S}!\mathsf{rep}\}$ and then, applying rule T-CH we eventually obtain a derivation of $\Pi' \vdash \mathsf{C} \overset{\mathsf{md}}{\dashrightarrow} \mathsf{S} + \mathsf{C} \overset{\mathsf{req}}{\dashrightarrow} \mathsf{S}; \mathsf{S} \overset{\mathsf{done}}{\dashrightarrow} \mathsf{C} : \langle \phi_1' \cup \phi_2', \Lambda_1' \cup \Lambda_3' \rangle$, that is the typing context in which we can safely replace $\mathsf{C} \xrightarrow{\mathsf{md}} \mathsf{B}; \mathsf{B} \xrightarrow{\mathsf{md}} \mathsf{S}$ to $\mathsf{C} \overset{\mathsf{md}}{\dashrightarrow} \mathsf{S}$, $\mathsf{C} \xrightarrow{\mathsf{x}} \mathsf{B}; \mathsf{B} \xrightarrow{\mathsf{req}} \mathsf{S}$ to $\mathsf{C} \overset{\mathsf{req}}{\dashrightarrow} \mathsf{S}$ and $\mathsf{S} \xrightarrow{\mathsf{done}} \mathsf{C}$ to $\mathsf{S} \overset{\mathsf{done}}{\dashrightarrow} \mathsf{C}$. ◇

5 Conclusions & Related Work

We proposed a framework for refining the global views of choreographies. In the context of concurrent and distributed systems, refinement methods have received great attention in the 80-90's. Action refinement has been studied in different settings by adding refinement combinators to, e.g., process algebras [1], labelled event structures [8] and causal trees [6]. The cornerstone in this line of work is that actions, considered atomic at a given level of abstraction, are refined into processes or computations, which are non-atomic at a lower level of abstraction. For instance, a labelled event structure can be refined into another one by substituting all events that have a particular label by an event structure [8]. Analogously, a term of a process algebra can be refined into another one by replacing all occurrences of a particular action by a term. A straightforward application of this approach to global choreographies would suggest to consider a standard language for choreographies, e.g., the one formalised in [14] and reproduced in Definition 3, and then provide a substitution mechanism for its atomic actions, which in this setting would be interactions $\mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B}$. Despite being technically possible, we opted for a generalised version $\mathsf{A} \overset{\mathsf{m}_1 \dots \mathsf{m}_n}{\dashrightarrow} \mathsf{B}_1 \dots \mathsf{B}_n$ that allows for the explicit definition of several participants involved in a complex, underspecified interaction. In this way, we provide more flexibility for abstraction. Suppose we would like to state that a participant A is intended to communicate two messages m_1 and m_2 respectively to B_1 and B_2 but A is uninterested on the way in which those messages are delivered, e.g., A is equally satisfied if m_1 arrives first to B_1 and then B_1 sends m_2 to B_2, or if the flow involves first B_2 and then B_1, or else if a (non-specified) broker sends the messages to both receivers. A specification of all behaviours described above in terms of binary interactions leaves little space for abstraction. This has been the main motivation for the introduction of refinable interactions. We may have opted for more general versions of refinable interactions, e.g., by considering multiple senders on the left hand side of the arrow. We opted for the current presentation because the standard well-formedness condition on global choreographies introduces several limitations on the way in which such abstractions could be implemented, e.g., they could not be refined as a choices because of the standard constraint about single selectors. We plan to investigate suitable generalisations on the shape of refinable interactions.

A semantics of g-choreographies in terms of pomsets [13] has been introduced in [9, 14]. Pomsets can be envisaged as event structures with an empty conflict relation. This semantics captures a more general notion of well-brancheness than the one considered here. In principle, one could borrow this more general notion of well-formedness in our framework at the cost of increasing the technical execution. Despite we associate global choreographies with event structures (as previously done, e.g., in [5] to give semantics to multi-party session types [10]), we remark that refinement techniques developed for event structures cannot be straightforwardly lifted to the language of global choreographies because the semantics of interactions is given in terms of two events. Hence, the refinement of an interaction would translate into the refinement of a sub-structure instead of a single event. This establishes an interesting connection with previous work that we plan to investigate further. Along the same lines, the main focus on previous work on refinement [1, 8, 6] is concerned with the consistency of refinement with respect to the semantics of the language, i.e., whether refinement preserves behavioural equivalences. Note that we have left implicit the semantics of refinable interactions, which is given in terms of the set of concrete realisations that it admits. An interesting line of work that we envisage for future development is whether the proposed refinement preserves equivalences.

The typing of ground g-choreography is unique (cf. Corollary 13). Actually, a simple inspection of the typing rules in Fig. 1 shows that type inference is trivial for ground g-choreography. This is not the case for non-ground g-choreographies where a type inference algorithm has to "guess" the types of refinable actions. We leave type inference open and we will address it in the future.

We also leave open the problem of which properties are maintained through the refinement process, namely what can be established of abstract programs that still holds of concrete ones. In particular, we would like to preserve well-formedness of g-choreographies along refinements.

References

[1] Luca Aceto & Matthew Hennessy (1994): *Adding action refinement to a finite process algebra*. Information and Computation 115(2), pp. 179–247, doi:10.1006/inco.1994.1096.

[2] Davide Basile, Pierpaolo Degano, Gian-Luigi Ferrari & Emilio Tuosto (2016): *Relating two automata-based models of orchestration and choreography*. JLAMP 85(3), pp. 425 – 446, doi:10.1016/j.jlamp.2015.09.011.

[3] Egon Börger (2012): *Approaches to modeling business processes: a critical analysis of BPMN, workflow patterns and YAWL*. Software and Systems Modeling 11(3), pp. 305–318, doi:10.1007/s10270-011-0214-z.

[4] Marco Carbone, Fabrizio Montesi & Hugo Torres Vieira (2018): *Choreographies for Reactive Programming*. CoRR abs/1801.08107. Available at http://arxiv.org/abs/1801.08107.

[5] Ilaria Castellani, Mariangiola Dezani-Ciancaglini & Paola Giannini (2019): *Event Structure Semantics for Multiparty Sessions*. In: Models, Languages, and Tools for Concurrent and Distributed Programming, LNCS 11665, Springer, pp. 340–363, doi:10.1007/978-3-030-21485-2_19.

[6] Philippe Darondeau & Pierpaolo Degano (1993): *Refinement of Actions in Event Structures and Causal Trees*. Theor. Comput. Sci. 118(1), pp. 21–48, doi:10.1016/0304-3975(93)90361-V.

[7] Edsger W. Dijkstra (1976): *A Discipline of Programming*. Prentice-Hall. Available at https://www.worldcat.org/oclc/01958445.

[8] Rob J. van Glabbeek & Ursula Goltz (1989): *Equivalence Notions for Concurrent Systems and Refinement of Actions (Extended Abstract)*. In: MFCS, LNCS 379, Springer, pp. 237–248, doi:10.1007/3-540-51486-4_71.

[9] Roberto Guanciale & Emilio Tuosto (2016): *An Abstract Semantics of the Global View of Choreographies*. In: ICE, EPTCS 223, pp. 67–82, doi:10.4204/EPTCS.223.5.

[10] Kohei Honda, Nobuko Yoshida & Marco Carbone (2016): *Multiparty Asynchronous Session Types*. Journal of the ACM 63(1), pp. 9:1–9:67, doi:10.1145/2827695. Extended version of a paper presented at POPL08.

[11] Nickolas Kavantzas, Davide Burdett, Gregory Ritzinger, Tony Fletcher & Yves Lafon (2004): *Web Services Choreography Description Language Version 1.0*. http://www.w3.org/TR/2004/WD-ws-cdl-10-20041217.

[12] Mogens Nielsen, Gordon D. Plotkin & Glynn Winskel (1981): *Petri Nets, Event Structures and Domains, Part I*. Theor. Comput. Sci. 13, pp. 85–108, doi:10.1016/0304-3975(81)90112-2.

[13] Vaughan R. Pratt (1986): *Modeling concurrency with partial orders*. Int. J. Parallel Program. 15(1), pp. 33–71, doi:10.1007/BF01379149.

[14] Emilio Tuosto & Roberto Guanciale (2018): *Semantics of global view of choreographies*. JLAMP 95, pp. 17 – 40, doi:10.1016/j.jlamp.2017.11.002.