Event-B-Supported Choreography-Defined Communicating Systems

Correctness and Completeness

Sarah Benyagoub[1], Yamine Aït-Ameur[1], and Klaus-Dieter Schewe[2](✉)

[1] Université de Toulouse, IRIT/INPT-ENSEEIHT, Toulouse, France
{sarah.benyagoub,yamine}@enseeiht.fr
[2] Zhejiang University, UIUC Institute, Haining, China
kd.schewe@intl.zju.edu.cn, kdschewe@acm.org

Abstract. Choreographies prescribe the rendez-vous synchronisation of messages in a communicating system. Such a system is called *realisable*, if the traces of the prescribed communication coincide with those of the asynchronous system of peers, where the communication channels either use FIFO queues or multiset mailboxes. It has recently been shown that realisability can be characterised by two necessary conditions that together are also sufficient, whereas in general the *synchronisability* of communicating peers is undecidable. The sufficiency of the conditions permits the construction of correct communicating systems; their necessity shows that all choreography-defined communicating system can be obtained in this way. This article provides an integrated framework based on Event-B for such a construction with a major emphasis on Rodin-based proofs of correctness and completeness.

Keywords: Event-B · Choreography · Realisability · Correctness proof · Completeness proof

1 Introduction

In a communicating system peers communicate asynchronously through messages. If the computations performed by the peers are disregarded and only the sequences of messages sent and received are considered, the system becomes a system of communicating FSMs with a semantics defined by the traces of sent messages. In addition, only those traces may be taken into account in which all sent messages have also been received.

Such a trace semantics can be defined in various ways using channels organised as FIFO queues for each pair of peers [6] or just for each receiver [1]. Alternatively, channels may be organised as multisets [8]. Naturally, one may also consider the possibility of messages being lost [7]. The *synchronisability* problem for such communicating systems is to decide whether the traces remain the same,

A. Raschke et al. (Eds.): ABZ 2020, LNCS 12071, pp. 155–168, 2020.
https://doi.org/10.1007/978-3-030-48077-6_11

if a rendez-vous synchronisation of sending and receiving of messages is considered. This was proven to be undecidable in general [9]. The picture changes in the presence of *choreographies* which prescribe the rendez-vous synchronisation [2]. In this case the peers are projections of a choreography, and synchronisability becomes *realisability* of the given choreography. Recently it was shown that in this case the rendez-vous composition of the projected peers coincides with the choreography, and *language synchronisability* based only on the message traces concides with *synchronisability* based in addition on the stable configurations reached [10]. This further enabled the characterisation of realisability of choreographies by two necessary conditions on a communication choreography, which together are sufficient.

A constructive Event-B-based approach to develop realisable choreographies and consequently communicating systems was brought up in [4,5]. The general idea is to exploit construction operators, by means of which realisable choreographies can be built out of a primitive base [11]. This already contains a hint on the sufficient conditions used in the associated proofs that were conducted using Rodin [3]. As the sufficiency proof in [10] removes some unnecessary assumptions, this approach becomes general. More importantly, the necessity of the conditions shows that all choreography-defined communicating systems can be obtained in this way. *In this paper we continue this route and show that also the necessity proof for realisable choreographies can be supported by Event-B and Rodin. This further gives us means for repairing choreographies.*

The remainder of this article is organised as follows. Section 2 is dedicated to theoretical foundations, where we review the fundamental definitions around peer-to-peer (P2P) systems and choreographies as well as the theory of realisable choreographies developed in [10]. Different to previous work we concentrate only on the most restrictive composition using a single message queue per peer. In Sect. 3 we briefly review our previous work on the Event-B-based construction of choreography-defined P2P systems with a slight extension of the Rodin-based proofs based on our newer insights. Section 4 is the core of this paper emphasising the necessary conditions for realisable choreographies and the Rodin-based proof. We conclude with a brief summary and outlook in Sect. 5.

2 Theoretical Background of Realisable Choreographies

Let M and P be finite, disjoint sets, elements of which are called *messages* and *peers*, respectively. Each message $m \in M$ has a unique *sender* $s(m) \in P$ and a unique *receiver* $r(m) \in P$ with $s(m) \neq r(m)$. We use the notation $i \xrightarrow{m} j$ for a message m with $s(m) = i$ and $r(m) = j$. We also use the notation $!m^{i \to j}$ and $?m^{i \to j}$ for the *event* of sending or receiving the message m, respectively. Write M_p^s and M_p^r for the sets of messages, for which the sender or the receiver is p, respectively.

Let $s(M)$ and $r(M)$ denote the sets of send and receive events defined by a set M of messages. A *P2P system* over M and P is a family $\{\mathcal{P}_p\}_{p \in P}$ of finite state machines (FSMs) $\mathcal{P}_p$ over an alphabet $\Sigma_p = s(M_p^s) \cup r(M_p^r)$. By abuse of terminology $\mathcal{P}_p$ is also called a *peer*.

We write $\mathcal{P}_p = (Q_p, \Sigma_p, q_{0,p}, F_p, \delta_p)$, where Q_p is the finite set of states of the FSM, $q_{0,p} \in Q_p$ is the start state, $F_p \subseteq Q_p$ is the set of final states, and δ_p is the transition function, i.e. $\delta_p : Q_p \times \Sigma_p \to Q_p$. Without loss of generality we may concentrate on deterministic FSMs (see [10, Prop.1]).

2.1 Composition of Peers

A composition of a P2P system over M and P will be another automaton, the alphabet of which will be either M or $s(M) \cup r(M)$.

The *rendez-vous composition* of a P2P system $\{\mathcal{P}_p\}_{1 \le p \le n}$ with $\mathcal{P}_p = (Q_p, \Sigma_p, q_{0p}, Q_p, \delta_p)$ is the FSM $\mathcal{C}_{rv} = (Q, M, q_0, Q, \delta)$ with $Q = Q_1 \times \cdots \times Q_n$, $q_0 = (q_{01}, \dots, q_{0n})$, and $\delta((q_1, \dots, q_n), i \xrightarrow{m} j) = (q_1', \dots, q_n')$ holds if $\delta_i(q_i, !m^{i \to j}) = q_i'$ and $\delta_j(q_j, ?m^{i \to j}) = q_j'$ hold, and $q_x = q_x'$ for all $x \notin \{i, j\}$.

The *mailbox composition* of a P2P system $\{\mathcal{P}_p\}_{1 \le p \le n}$ with $\mathcal{P}_p = (Q_p, \Sigma_p, q_{0p}, Q_p, \delta_p)$ is the automaton $\mathcal{C}_m = (Q, \Sigma, q_0, Q, \delta)$ satisfying the following conditions:

- The set of states is $Q = Q_1 \times \cdots \times Q_n \times (c_j)_{1 \le j \le n}$, where each c_j is a finite queue with elements in M.
- The alphabet is $\Sigma = s(M) \cup r(M)$.
- The initial state is $q_0 = (q_{01}, \dots, q_{0n}, ([])_{1 \le j \le n})$, i.e. initially all channels are empty.
- The transition function δ is defined by $\delta((q_1, \dots, q_n, (c_j)_{1 \le j \le n}), e) = (q_1', \dots, q_n', (c_j')_{1 \le j \le n})$ if there exists i such that $\delta_i(q_i, e) = q_i'$ holds, $q_x = q_x'$ for all $x \neq i$, and
  - either $e = !m^{i \to j}$ for some j, $c_j' = c_j {}^\frown [i \xrightarrow{m} j]$, and $c_k = c_k'$ for all $k \neq j$
  - or $e = ?m^{j \to i}$ for some j and $c_i = [j \xrightarrow{m} i] {}^\frown c_i'$ and $c_k = c_k'$ for all $k \neq i$.

As above we call a state $(q_1, \dots, q_n, (c_j)_{1 \le j \le n})$ *stable* if and only if all channels c_j are empty.

Peers as well as any composition of a P2P system are defined by automata, so their semantics is well defined by the notion of language accepted by them. It is common to consider just sequences of sending events, i.e. for a word $w \in M^*$ let $\sigma(w)$ denote its restriction to its sending events. Formally, we have $\sigma(\epsilon) = \epsilon$, $\sigma(i \xrightarrow{m} j) = !m^{i \to j}$, and $\sigma(w_1 \cdot w_2) = \sigma(w_1) \cdot \sigma(w_2)$, where $\cdot$ denotes concatenation. Analogously, for words in $(s(M) \cup r(M))^*$ we have $\sigma(\epsilon) = \sigma(?m^{i \to j}) = \epsilon$, $\sigma(!m^{i \to j}) = !m^{i \to j}$, and $\sigma(w_1 \cdot w_2) = \sigma(w_1) \cdot \sigma(w_2)$.

If $\mathcal{L}$ is the language accepted by an FSM $\mathcal{A}$ with alphabet M or $\Sigma = s(M) \cup r(M)$, then $\mathcal{L}(\mathcal{A}) = \sigma(\mathcal{L})$ is the *trace language* of $\mathcal{A}$. This applies for the cases where $\mathcal{A}$ is a peer $\mathcal{P}_p$ or a composition $\mathcal{C}_{rv}$ or $\mathcal{C}_m$. We use the notation $\mathcal{L}_0(\mathcal{P}) = \mathcal{L}(\mathcal{C}_{rv})$, $\mathcal{L}_\omega(\mathcal{P}) = \mathcal{L}(\mathcal{C}_m)$.

If we restrict final states to be stable, we obtain a different language $\hat{\mathcal{L}}(\mathcal{C}_m) \subseteq \mathcal{L}(\mathcal{C}_m)$, which we call the *stable trace language* of $\mathcal{C}_m$.

A P2P system $\mathcal{P} = \{\mathcal{P}_p\}_{1 \le p \le n}$ is called *language-synchronisable*, if $\mathcal{L}_0(\mathcal{P}) = \mathcal{L}_\omega(\mathcal{P})$ holds. $\mathcal{P} = \{\mathcal{P}_p\}_{1 \le p \le n}$ is called *synchronisable*, if $\mathcal{L}_0(\mathcal{P}) = \mathcal{L}_\omega(\mathcal{P}) = \hat{\mathcal{L}}_\omega(\mathcal{P})$ holds.

2.2 Choreography-Defined P2P Systems

Let us now look into choreographies. We define a *choreography* by an FSM $\mathcal{C} = (Q, M, q_0, F, \delta)$, where M is again a set of messages. As before we ignore final states and assume $F = Q$. Then every rendez-vous composition of a P2P system $\mathcal{P} = \{\mathcal{P}_p\}_{1 \le p \le n}$ defines a choreography.

Let $\mathcal{C} = (Q, M, q_0, Q, \delta)$ be a choreography with messages M and peers P. For $p \in P$ the *projection* $\pi_p(\mathcal{C})$ is the FSM $(Q, \Sigma, q_0, Q, \delta_p)$ with $\Sigma = s(M) \cup r(M)$ and $\delta_p(q, e) = q'$ if $e = !m^{p \to j}$ for some j with $\delta(q, p \xrightarrow{m} j) = q'$, $e = ?m^{i \to p}$ for some i with $\delta(q, i \xrightarrow{m} p) = q'$ or $e = \epsilon$ for $\delta(q, i \xrightarrow{m} j) = q'$ with $p \notin \{i, j\}$.

The *peer* $\mathcal{P}_p$ *defined by* $\mathcal{C}$ is the FSM without ϵ-transitions corresponding to $\pi_p(\mathcal{C})$. A P2P system $\mathcal{P} = \{\mathcal{P}_p\}_{1 \le p \le n}$ is *choreography-defined* if there exists a choreography with peers $\mathcal{P}_p$ for all p.

There is a close relationship between rendez-vous compositions and choreography-defined P2P systems. In [10] we proved that each choreography $\mathcal{C}$ coincides (up to isomorphism) with the rendez-vous composition of its peers. Thus, not all P2P systems are choreography-defined. In fact, if a P2P system is choreography-defined, then it must consist of the peers defined by its rendez-vous composition.

For choreography-defined P2P systems the synchronisability problem is much simpler than in the general case. In [10] we proved that a choreography-defined P2P system $\mathcal{P} = \{\mathcal{P}_p\}_{1 \le p \le n}$ is synchronisable if and only if it is language-synchronisable.

Therefore, we may focus only on language-synchronisability: if a trace is accepted, then it will be accepted in a stable configuration. We may also identify the rendez-vous composition with the given choreography. Therefore, a choreography $\mathcal{C}$ is called *realisable*, if $\mathcal{L}_0(\mathcal{P}) = \mathcal{L}_\omega(\mathcal{P})$ holds for the P2P system $\mathcal{P}$ defined by the projections of $\mathcal{C}$.

2.3 Characterisation of Realisability

The main result from [10] states that there are two necessary conditions for realisability, which together are sufficient. The sequence condition expresses that if two messages appear in a sequence, the sender of the second message must coincide with either the sender or the receiver of the preceding message. The choice condition expresses that if there is a choice of continuation with two different messages, then these messages must have the same sender.

Sequence Condition. Whenever there are states $q_1, q_2, q_3 \in Q$ with $\delta(q_1, i \xrightarrow{m_1} j) = q_2$ and $\delta(q_2, k \xrightarrow{m_2} \ell) = q_3$ for non-independent messages $i \xrightarrow{m_1} j$ and $k \xrightarrow{m_2} \ell$, we must have $k \in \{i, j\}$.

Choice Condition. Whenever there are states $q_1, q_2, q_3 \in Q$ with $\delta(q_1, i \stackrel{m_1}{\to} j) = q_2$, $\delta(q_1, k \stackrel{m_2}{\to} \ell) = q_3$ and $q_2 \neq q_3$ for non-independent messages $i \stackrel{m_1}{\to} j$ and $k \stackrel{m_2}{\to} \ell$, we must have $k = i$.

Both conditions establish constraints on δ for two messages $i \stackrel{m_1}{\to} j$ and $k \stackrel{m_2}{\to} \ell$, but in both cases we need to exclude that these two messages are *independent* in the sense that they may appear in any order, i.e. we request that if there are states q_1, q_2, q_3 with $\delta(q_1, i \stackrel{m_1}{\to} j) = q_2$ and $\delta(q_2, k \stackrel{m_2}{\to} \ell) = q_3$, then we cannot have both $\delta(q_1, k \stackrel{m_2}{\to} \ell) = q_2$ and $\delta(q_2, i \stackrel{m_1}{\to} j) = q_3$. The following theorem is the main result in [10].

Theorem 1. *A choreography $\mathcal{C}$ is a realisable with respect to P2P, queue or mailbox composition if and only if it satisfies the sequence and choice conditions.*

3 Correctness by Construction

We now address the construction of realisable choreographies. For this we will first introduce several composition operators in Subsect. 3.1. We can easily define conditions on the constructors to ensure that all choreographies obtained by composition will satisfy the choice and sequence conditions and thus are realisable by Theorem 1. However, following [3,4] we will actually redo the (sufficiency) proof using specifications of the constructors in Event-B and the Rodin prover.

3.1 Composition Operators

In the following we use the notation CP to refer to a choreography, and we add indices to distinguish different choreographies, whenever the need arises. Without loss of generality we also introduce distinguished final states q^f_{CP}, which ease the proofs. We define three composition operators:*sequence* composition $\otimes_{(\gg, q^f_{CP})}$, *branching* composition $\otimes_{(+, q^f_{CP})}$, and *loop* composition $\otimes_{(\circlearrowleft, q^f_{CP})}$.

Each expression of the form $\otimes_{(op, q^f_{CP})}(CP, CP_b)$ assumes that the initial state of CP_b is fused with the final state s^f_{CP}. Informally, we can say that CP_b is appended to CP at state s^f_{CP}.

Definition 1 (Sequential Composition). Given a choreograhy CP with final state $q_{CP} \in Q^f_{CP}$ and a choreography CP_b with a single transition $\delta_{CP_b}(q_{CP_b}, l_{CP_b}) = q'_{CP_b}$, the *sequential* composition $CP_{\gg} = \otimes_{(\gg, s_{CP})}(CP, CP_b)$ is defined by $Q_{CP_{\gg}} = Q_{CP} \cup \{q'_{CP_b}\}$, $M_{CP_{\gg}} = M_{CP} \cup \{m_{CP_b}\}$, $Q^f_{CP_{\gg}} = (Q^f_{CP} \setminus \{q_{CP}\}) \cup \{q'_{CP_b}\}$ and $\delta_{CP_{\gg}} = \delta_{CP} \cup \{((q_{CP}, l_{CP_b}), q'_{CP_b})\}$.

Definition 2 (Branching Composition). Given a choreography CP with final state $q_{CP} \in Q^f_{CP}$ and a family of choreographies $\{CP_{bi}\}_{1 \leq i \leq n}$, each comprising a single transition $\delta_{CP_{bi}}(q_{CP_{bi}}, l_{CP_{bi}}) = q'_{CP_{bi}}$, the *branching* composition $CP_+ = \otimes_{(+, q_{CP})}(CP, \{CP_{bi}\})$ is defined by

- $Q_{CP_+} = Q_{CP} \cup \{q'_{CP_1}, \ldots, q'_{CP_{bn}} \mid \delta_{CP_{bi}}(q_{CP_{bi}}, l_{CP_{bi}}) = q'_{CP_{bi}}\}$,
- $M_{CP_+} = M_{CP} \cup \{l_{CP_{bi}}, \ldots, l_{CP_{bn}}\}$
- $\delta_{CP_+} = \delta_{CP} \cup \{((q_{CP}, l_{CP_{bi}}), q'_{CP_{bi}}) \mid 1 \leq i \leq n\}, and$
- $Q^f_{CP_+} = (Q^f_{CP} \setminus \{q_{CP}\}) \cup \{q'_{CP_{b1}}, \ldots, q'_{CP_{bn}}\}$.

Definition 3 (Loop Composition). Given a choreography CP with final state $q_{CP} \in Q^f_{CP}$ and a choreography CP_b with a single transition $\delta_{CP_b}(q_{CP_b}, l_{CP_b}) = q'_{CP_b}$ such that $q'_{CP_b} \in Q_{CP}$ holds, the *loop* composition $CP_{\circlearrowleft} = \otimes_{(\circlearrowleft, s_{CP})}(CP, CP_b)$ is defined by $Q_{CP_{\circlearrowleft}} = Q_{CP}$, $M_{CP_{\circlearrowleft}} = M_{CP} \cup \{l_{CP_b}\}$, $\delta_{CP_{\circlearrowleft}} = \delta_{CP} \cup \{((q_{CP}, l_{CP_b}), q'_{CP_b})\}$, and $Q^f_{CP_{\circlearrowleft}} = Q^f_{CP}$.

Clearly, according to Theorem 1 we must require that in a sequence the sender of the added message equals the sender or receiver of any message associated with a transition to $q_{CP} \in Q^f_{CP}$. The same must hold for the new messages introduced by a branching composition. In addition, the senders associated with the new messages must be pairwise different. In case of a loop composition we must in addition require that the sender of any message associated with a transition from $q'_{CP_b} \in Q_{CP}$ equals the sender or receiver of the newly introduced message.

3.2 Correctness Proof

We use Event-B to prove the correctness of the compositions thereby giving an alternative Rodin-based proof of Theorem 1. An Event-B model (see Table 1) is defined to encode this incremental process.

Table 1. An excerpt of the LTS_model.

```
INITIALISATION≜
EVENTS
    Add_Seq ≜
        Any Some_cp_b
        Where
            grd1: Some_cp_b ∈ cps_b
            grd2: MESSAGE(Some_cp_b) ≠ End
            grd3: Some_cp_b ∈ ISeqF
            grd4: SOURCE_STATE(Some_cp_b) ∈ CP_Final_states
            ...
        Then
            act1: BUILT_CP := BUILT_CP ∪ {Some_cp_b}
            act3: CP_Final_states := (CP_Final_states ∪
                      {DESTINATION_STATE(Some_cp_b)})\
                      {SOURCE_STATE(Some_cp_b)}
            ...
    End
    Add_Choice ≜ ...
    Add_Loop ≜ ...
    Add_End ≜ ...
End
```

Once initialisation ($INITIALISATION$) is performed, three events (*Add_Sequence*, *Add_Choice* and *Add_Loop*) for sequence, choice and loop are interleaved to build a choreography CP. All these events are guarded by the identified

Table 2. An excerpt of the LTS_CONTEXT.

```
LTS_CONTEXT
SETS PEERS, MESSAGES , CP_STATES.
CONSTANTS CPs_B, DC, ISeqF, NDC, ...
AXIOMS
    axm1: CPs_B ⊆ CP_STATES × PEERS × MESSAGES× PEERS × CP_STATES×ℕ
        − Determinstic CP definition DC
    axm2_Cond1: NDC ⊆ CPs_B
    axm3_Cond1: ∀Trans2, Trans1·(Trans1 ∈ CPs_B ∧ Trans2 ∈ CPs_B∧
                 SOURCE_STATE(Trans1) = SOURCE_STATE(Trans2)∧
                 LABEL(Trans1) = LABEL(Trans2)∧
                 DESTINATION_STATE(Trans1) ≠ DESTINATION_STATE(Trans2))
                         ⇒{Trans1, Trans2} ⊆ NDC
    axm4_Cond1: DC = CPs_B \ NDC
        − Independent sequence freeness definition ISEQF
    axm5_Cond2: ISeqF ⊆ CPs_B
    axm6_Cond2: ∀ cp_b · ( cp_b ∈ CPs_B ∧
                 (PEER_SOURCE(cp_b) = LAST_SENDER_PEERS(SOURCE_STATE(cp_b)) ∨
                 PEER_SOURCE(cp_b) = LAST_RECEIVER_PEERS(SOURCE_STATE(cp_b))))
                         ⇒  {cp_b} ⊆ ISeqF
        − Parallel Choice freeness PCF
    axm7_Cond3: PCF ⊆ CPs_B
    axm8_Cond3: ∀ cp_b· (cp_b ∈ CPs_B ∧
                 {PEER_SOURCE(cp_b)} = BRANCHES_PEERS_SOURCE(cp_b) )
                 ⇒ {cp_b} ⊆ PCF
    ...
End
```

conditions deterministic, sequence and choice conditions defined in the context $LTS\_CONTEXT$ of Table 2.

In this context (see Table 2), we introduce using sets and constants, the whole basic definitions of messages, choreography states, basic choreographies (i.e. choreographies with a single transition as used in the definitions of the composition operators), etc. A set of axioms is used to define the relevant properties of these definitions. For example, in Axiom $axm1$, a choreography CP is defined as a set of transitions with a source and target state, a message and a source and target peers. Axiom $axm3\_Cond1$ defines that a non-deterministic choreography is using the NDC set. This NDC set characterises all the non-deterministic choices in a choreography CP. Note that Axiom $axm4\_Cond1$ defines the assumed deterministic choice condition. The capture of sequence conditions is given by Axioms $axm5\_Cond2$ and $axm6\_Cond2$. It compares the source peer $PEER\_SOURCE(cp\_b)$ with the sender peer $LAST\_SENDER\_PEERS$ or with the receiver peer $LAST\_RECEIVER\_PEERS$ of the last transition of the choreography.

Similarly, to define the choice condition, in Axioms $axm7\_Cond3$ and $axm8\_Cond3$ the sender peers $PEER\_SOURCE(Trans)$ of the transitions involved in a branch are compared.

The correctness proof rebuilds the three decisive parts of the proof of Theorem 1:

1. It shows that the trace language of the choreography coincides with the one of the rendez-vous composition of its projected peers. This property was called *equivalence* in earlier work [2].

2. It shows the language synchronisability between the rendez-vous composition and the mailbox composition, which was referred to as *synchonisability* in [2].
3. It shows that all accepted sequence of messages of the mailbox composition system are accepted in a state, where the mailboxes are empty. This property was called *well-formedness* in earlier work [2].

4 Completeness Proof: A Correct-by-Construction Approach with Event-B

To prove that all the choreographies CP built using the previously defined events, encoding the composition operators, we rely on refinement offered by Event-B. As indicated above we can decomposed the realisability property into three properties, namely *equivalence*, *synchronisability* and *well-formedness*. The Event-B context in Table 3 defines these properties.

Table 3. An excerpt of the LTS_SYNC_CONTEXT.

```
LTS_SYNC_CONTEXT, EXTENDS LTS_CONTEXT
SETS ACTIONS. CONSTANTS CPs_B , EQUIV, . . .
AXIOMS
    axm1: CPs_SYNC_B ⊆ CP_STATES × ACTIONS × MESSAGES × PEERS×
                        PEERS × ACTIONS × MESSAGES × CP_STATES × ℕ
    axm2: CPs_ASYNC_B ∈ (A_STATES × ETIQ × ℕ) ⇸ A_STATES
        – Equivalence of CP and Synchronous projection
    axm_1.a: EQUIV ∈ CPs_B ⤖ CPs_SYNC_B
    axm_1.a1: EQUIV = { Trans ↦ S_Trans | Trans ∈ CPs_B ∧ S_Trans ∈ CPs_SYNC_B ∧
                SOURCE_STATE(Trans) = S_SOURCE_STATE(S_Trans) ∧
                DESTINATION_STATE(Trans) = S_DESTINATION_STATE(S_Trans) ∧
                PEER_SOURCE(Trans) = S_PEER_SOURCE(S_Trans) ∧
                PEER_DESTINATION(Trans) = S_PEER_DESTINATION(S_Trans) ∧
                MESSAGE(Trans) = S_MESSAGE(S_Trans) ∧
                INDEX(Trans) = S_INDEX(S_Trans) }
        – Synchronisability property
    axm_1.b:  SYNCHRONISABILITY  ∈ CPs_SYNC_B ⤖ R_TRACE_B
    axm_1.b1:  SYNCHRONISABILITY  = {S_Trans↦ R_Trans | S_Trans ∈ CPs_SYNC_B ∧
            R_Trans ∈ R_TRACE_B ∧ S_INDEX(S_Trans) = R_INDEX(R_Trans) ∧
            S_SOURCE_STATE(S_Trans) = R_SOURCE_STATE(R_Trans) ∧
            S_PEER_SOURCE(S_Trans) = R_PEER_SOURCE(R_Trans) ∧
            S_MESSAGE(S_Trans) = R_MESSAGE(R_Trans) ∧
            S_PEER_DESTINATION(S_Trans) = R_PEER_DESTINATION(R_Trans) ∧
            S_DESTINATION_STATE(S_Trans) = R_DESTINATION_STATE(R_Trans)}
        – Well formedness property
    axm_1.c: WF  ∈ A_TRACES → QUEUE
    axm_1.c1: ∀ A_TR,queue · ( A_TR ∈ A_TRACES ∧ queue ∈ QUEUE ∧ queue = ∅ )
     ⇒ A_TR↦ queue ∈  WF
     . . .
End
```

Each property is formalised by a set of choreographies satisfying the corresponding property. These definitions use the rendez-vous composition CP_{rv} defined as set CPs_SYNC_B and the mailbox composition CP_m defined as set CPs_ASYNC_B in context $LTS\_SYNC\_CONTEXT$ of Table 3.

4.1 An Event-B Context for the Realisability Property

The definition of the state-transitions system corresponding to the synchronous projection is given by the set CPs_SYNC_B defined by Axiom $axm1$ in Table 3. Actions (send ! and receive ?) are introduced. Then two other important axioms, namely $axm\_1.a$ and $axm\_1.a1$, are given to define the equivalence between a choreography CP and its synchronous projection. The $EQUIV$ relation is introduced. It characterises the set of CPs that are equivalent to their synchronous projection. Axiom $axm\_1.a1$ formalises *equivalence*. The properties related to *synchronisability* are captured by Axioms $axm\_1.b$ and $axm\_1.b1$. Well-formedness is captured by Axioms $axm\_1.c$ and $axm\_1.c1$.

4.2 Refinement

We exploit the characterisation of realisability by three properties in a refinement strategy, which establishes the necessity step-by-step. These properties are introduced as invariants and inductively proven for each composition operator (sequence, choice and loop). That is, two refinements of the initial machine of Table 1 are defined:

- The first refinement introduces the equivalence property by defining the (synchronous) rendez-vous projection of the initial choreography CP.
- Synchronisability and well-formedness properties are proven in the second refinement.

Below we present a sketch of this development focusing on the definition of the sequence operator. The complete development can be accessed from http://yamine.perso.enseeiht.fr/ABZ2020EventBModels.pdf.

Table 4. An excerpt of the LTS_Synchronous_model.

INITIALISATION
    ...
EVENTS
    Add_Seq **Refines** $Add\_Seq \triangleq$
        Any
            $S\_Some\_cp\_b, Some\_cp\_sync\_b$
        Where
            grd1: $Some\_cp\_sync\_b \in cps\_sync\_b$
            grd3: $S\_SOURCE\_STATE(Some\_cp\_sync\_b) \in CP\_Final\_states$
            grd4: $S\_Some\_cp\_b \in ISeq$
            grd8: $MESSAGE(S\_Some\_cp\_b) \neq End$
            grd9: $MESSAGE(S\_Some\_cp\_b) = S\_MESSAGE(Some\_cp\_sync\_b)$
            ...
        With Some_cp_b: $Some\_cp\_b = S\_Some\_cp\_b$
        Then
            act1: $BUILT\_CP := BUILT\_CP \cup \{S\_Some\_cp\_b\}$
            act2: $BUILT\_SYNC := BUILT\_SYNC \cup \{Some\_cp\_sync\_b\}$
        ...
    End

First Refinement: Equivalence. The first refinement introduces the synchronous projection of the $BUILT\_CP$ defined by variable $BUILT\_SYNC$ in Table 4.

The event *Add_Seq* or sequence operator (Table 4) refines the same event of the root model of Table 1. It introduces the $BUILT\_SYNC$ set corresponding to the synchronous projection as given in Sect. 2.1. Here, again, the *Add_Seq* applies only if the conditions in the guards hold. The $With$ clause provides a witness to glue *Some_cp_b* CP with its synchronous version.

Second Refinement: Synchronisability and Well-Formedness. The second refinement introduces the asynchronous projection with sending and receiving peers actions.

Well-formedenss and synchronisability remain to be proven in order to complete realisability preservation. At this level each event corresponding to a composition operator is refined by three events: one to handle sending of messages (*Add_Seq_send*), one for receiving messages (*Add_Seq_receive*), and a third one (*Add_Seq_send_ receive*) refining the abstract *Add_seq* event. Queues are introduced as well.

Table 5 defines these events. Sending and receiving events are interleaved in an asynchronous manner. Once a pair of send and receive events has been triggered, the event *Add_Seq_send_receive* records that the emission-reception is completed. This event increases the number of received messages (Action $act5$). Traces are updated accordingly by the events, they are used for proving the invariants.

4.3 Completeness Proof

The proof of completeness consists in proving a choreography is realisable if and only if it is built using the defined composition operators. The Rodin-based proofs exploits that realisability can be equivalently expressed by *equivalence*, *synchronisability* and *well-formedness*. Note that the proof strategy with the sufficiency and necessity parts is quite similar, as the same development and refinement steps are used in both cases. The main difference resides in the definition of two invariants, which correspond to each direction of the implication corresponding to the necessity and sufficiency conditions.

Sufficiency. Sufficiency consists in proving that, if a choreography is built using the defined composition operator, then it is realisable. This property has been proven by proving the invariants described in Table 6.

These invariants state that for each CP built using the composition operators, the obtained CP fulfils *Equivalence*, *Synchronisability* and *WF* by set belonging property. Table 6 introduces the *equivalence* property through invariant `inv_1.a`. The invariant requires equivalence between a CP and its synchronous projection. `inv_1.b` and `inv_1.c` introduce respectively the *synchronisability* and *well-formedness* properties.

Table 5. An excerpt of the LTS_Asynchronous_model.

Event $Add\_Seq\_Send \triangleq$
Any
$send, lts\_s, lts\_d, msg, index$
Where
grd1: $\exists send\_st\_src, send\_st\_dest \cdot ((lts\_s \mapsto send\_st\_src) \in A\_GS \wedge ((send\_st\_src \mapsto (Send \mapsto msg \mapsto lts\_d) \mapsto index) \mapsto send\_st\_dest) \in CPs\_ASYNC\_B \wedge \ldots$
...
Then
act1: $A\_TRACE := A\_TRACE \cup \{Reduces\_Trace\_states \mapsto St\_Num \mapsto Send \mapsto lts\_s \mapsto msg \mapsto lts\_d \mapsto Reduces\_Trace\_states \mapsto (St\_Num + 1) \mapsto A\_Trace\_index\}$
act2: $queue, back := queue \cup \{lts\_d \mapsto msg \mapsto back\}, back + 1$
act3: $A\_GS := A\_Next\_States(\{send\} \mapsto A\_GS \mapsto queue)$
...
End
Event $Add\_Seq\_Receive \triangleq$
Any
$send, receive, lts\_s, lts\_d, msg, index$
Where
grd1: $queue \neq \varnothing \wedge lts\_d \mapsto msg \mapsto front \in queue$
grd2: $\exists receive\_st\_src, receive\_st\_dest \cdot (((lts\_d \mapsto receive\_st\_src) \in A\_GS) \wedge ((receive\_st\_src \mapsto (Receive \mapsto msg \mapsto lts\_s) \mapsto index) \mapsto receive\_st\_dest) \in CPs\_ASYNC\_B \wedge \ldots$
...
Then
act1: $A\_TRACE := A\_TRACE \cup \{Reduces\_Trace\_states \mapsto St\_Num \mapsto Receive \mapsto lts\_s \mapsto msg \mapsto lts\_d \mapsto Reduces\_Trace\_states \mapsto (St\_Num + 1) \mapsto A\_Trace\_index\}$
act2: $queue := queue \setminus \{lts\_d \mapsto msg \mapsto front\}$
...
End
Event $Add\_Seq\_Send - Receive$ **Refines** $Add\_Seq \triangleq$
Any
$A\_Some\_cp\_b, A\_Some\_cp\_sync\_b, Send\_cp\_async\_b, Receive\_cp\_async\_b, R\_trace\_b$
Where
grd1: $A\_MESSAGE(Send\_cp\_async\_b) = A\_MESSAGE(Receive\_cp\_async\_b)$
grd2: $ACTION(Receive\_cp\_async\_b) = Receive \wedge ACTION(Send\_cp\_async\_b) = Send$
grd3: $A\_Some\_cp\_b \in ISeq$
grd4: $MESSAGE(A\_Some\_cp\_b) = A\_MESSAGE(Send\_cp\_async\_b)$
...
With $S\_Some\_cp\_b : S\_Some\_cp\_b = A\_Some\_cp\_b,$
$Some\_cp\_sync\_b : Some\_cp\_sync\_b = A\_Some\_cp\_sync\_b$
Then
act1: $BUILT\_CP := BUILT\_CP \cup \{A\_Some\_cp\_b\}$
act2: $BUILT\_SYNC := BUILT\_SYNC \cup \{A\_Some\_cp\_sync\_b\}$
act3: $BUILT\_ASYNC := BUILT\_ASYNC \cup \{Send\_cp\_async\_b\} \cup \{Receive\_cp\_async\_b\}$
act4: $REDUCED\_TRACE := REDUCED\_TRACE \cup \{R\_trace\_b\}$
...
End
...
End

Necessity. Necessity consists in proving that if a CP is realisable, then it is built using the defined composition operator.

This property has been established by proving the invariants described in Table 7. Invariant `inv2.a` states that any CPbelonging to the *equivalence* set is a peer to peer CP, `inv2.b` states that any synchronisable CP belongs to the set of built CP and finally `inv2.c` states that all the well formed CP exchanging the ending message is built at the asynchronous level.

Proof Statistics. Table 8 gives the results of our experiments. We can observe that all the proof obligations (POs) have been proved. A large amount of these

Table 6. An excerpt of sufficient model invariants.

Invariants

inv1: $BUILT\_SYNC \subseteq CPs\_SYNC\_B$
inv2 $BUILT\_ASYNC \subseteq CP\_ASYNC\_B$
inv3 $REDUCED\_TRACE \subseteq R\_TRACE\_B$
inv4 $A\_TRACE \subseteq A\_TRACES$
inv_1.a: $\forall Trans \cdot \exists S\_Trans \cdot (Trans \in BUILT\_CP \wedge S\_Trans \in BUILT\_SYNC \wedge BUILT\_CP \neq \varnothing) \Rightarrow \mathbf{Trans} \mapsto \mathbf{S\_Trans} \in \mathbf{EQUIV}$
inv_1.b $\forall S\_Trans \cdot \exists R\_Trans \cdot (S\_Trans \in BUILT\_SYNC \wedge R\_Trans \in REDUCED\_TRACE) \Rightarrow \mathbf{S\_Trans} \mapsto \mathbf{R\_Trans} \in \mathbf{SYNCHRONISABILITY}$
inv_1.c $\forall A\_Trans \cdot (A\_Trans \in A\_TRACES \wedge MESSAGE(Last\_cp\_trans) = End \wedge A\_TRACE \neq \varnothing) \Rightarrow \mathbf{A\_Trans} \mapsto \mathbf{queue} \in \mathbf{WF}$
...

Table 7. An excerpt of necessary and sufficient model invariants.

Invariants

inv2.a $\forall Trans.\exists S\_Trans.(\mathbf{Trans} \mapsto \mathbf{S\_Trans} \in \mathbf{EQUIVALENCE} \wedge BUILT\_CP \neq \varnothing) \Rightarrow Trans \in BUILT\_CP \wedge S\_Trans \in BUILT\_SYNCHRONE$
inv2.b $\forall S\_Trans.\exists R\_Trans.(\mathbf{S\_Trans} \mapsto \mathbf{R\_Trans} \in \mathbf{SYNCHRONISABILITY} \wedge BUILT\_SYNCHRONE \neq \varnothing \wedge REDUCED\_TRACE \neq \varnothing) \Rightarrow S\_Trans \in BUILT\_SYNCHRONE \wedge R\_Trans \in REDUCED\_TRACE$
inv2.c $\forall A\_Trans.(\mathbf{A\_Trans} \mapsto \mathbf{queue} \in \mathbf{WF}) \Rightarrow (A\_Trans \in A\_TRACES \wedge queue = \varnothing \wedge MESSAGE(Last\_cp\_trans) = End\_message)$
...

Table 8. Rodin proofs statistics

Event-B model	Interactive proofs	Automatic proofs	Proof Obligations
Abstract context	06 (100%)	0 (0%)	06 (100%)
Synchronous context	02 (100%)	0 (0%)	02 (100%)
Asynchronous context	01 (33,33%)	02 (66,67%)	03 (100%)
Abstract model	28 (58,33%)	20 (41,67%)	48 (100%)
Synchronous model	43 (41,34%)	61 (58,65%)	104 (100%)
Asynchronous model	81 (41,32%)	115 (58,67%)	196 (100%)
Total	161 (100%)	198 (100%)	359 (100%)

POs has been proved automatically using the different provers associated to the Rodin platform. Interactive proofs of POs required to combine some interactive deduction rules and the automatic provers of Rodin. Few steps were required in most of the cases, and a maximum of 15 steps was reached.

5 Conclusion

In this article we extended the Event-B-based approach to the construction of realisable choreographies [4,5] based on recent new insights into choreography-defined P2P systems. In [10] we proved that under the presence of a choreography that prescribes the rendez-vous synchronisation of the peers there are two necessary conditions on realisable choreographies which together guarantee realisability. A consequence is decidability of realisability in the presence of a choreography. We removed unnecessary assumptions in the Event-B-based proofs and extended them to cover also necessity of the conditions. In doing so we demonstrated the power of the Rodin tool. All the models are accessible through http://yamine.perso.enseeiht.fr/ABZ2020EventBModels.pdf.

Naturally, using Event-B in this context provides an open invitation for a refinement-based approach taking choreographies to communicating systems that do not just emphasise the flow of messages. As we are now able to detect violations of a necessary condition, it allows us to find minimal repairs to the choreography to restore realisability. Such repairs have to be validated by a designer. In addition, we need a systematic investigation of refinements based on Event-B. In this context an analysis of the realisation of the messaging channels is due, for which we expect the most natural semantics using mailboxes to be the simplest to be realised. This refinement method provides an open invitation for the continuation of this research towards a verifiable method for the specification and refinement of correct P2P systems.

References

1. Basu, S., Bultan, T.: On deciding synchronizability for asynchronously communicating systems. Theor. Comput. Sci. **656**, 60–75 (2016)
2. Basu, S., Bultan, T., Ouederni, M.: Deciding choreography realizability. In: Field, J., Hicks, M. (eds.) Proceedings of the 39th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2012), pp. 191–202. ACM, New York (2012)
3. Benyagoub, S., Aït-Ameur, Y., Ouederni, M., Mashkoor, A., Medeghri, A.: Formal design of scalable conversation protocols using Event-B: validation, experiments and benchmarks. J. Softw.: Evol. Process **23**(2), 129–145 (2019)
4. Benyagoub, S., Ouederni, M., Aït-Ameur, Y., Mashkoor, A.: Incremental construction of realizable choreographies. In: Dutle, A., Muñoz, C., Narkawicz, A. (eds.) NFM 2018. LNCS, vol. 10811, pp. 1–19. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-77935-5_1
5. Benyagoub, S., Ouederni, M., Singh, N.K., Ait-Ameur, Y.: Correct-by-construction evolution of realisable conversation protocols. In: Bellatreche, L., Pastor, Ó., Almendros Jiménez, J.M., Aït-Ameur, Y. (eds.) MEDI 2016. LNCS, vol. 9893, pp. 260–273. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45547-1_21
6. Brand, D., Zafiropulo, P.: On communicating finite-state machines. J. ACM **30**(2), 323–342 (1983)
7. Chambart, P., Schnoebelen, P.: Mixing lossy and perfect FIFO channels. In: van Breugel, F., Chechik, M. (eds.) CONCUR 2008. LNCS, vol. 5201, pp. 340–355. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85361-9_28
8. Clemente, L., Herbreteau, F., Sutre, G.: Decidable topologies for communicating automata with FIFO and bag channels. In: Baldan, P., Gorla, D. (eds.) CONCUR 2014. LNCS, vol. 8704, pp. 281–296. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44584-6_20

9. Finkel, A., Lozes, É.: Synchronizability of communicating finite state machines is not decidable. In: Chatzigiannakis, I., et al. (eds.) 44th International Colloquium on Automata, Languages, and Programming (ICALP 2017), volume 80 of LIPIcs, pp. 122:1–122:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2017)
10. Schewe, K.-D., Aït-Ameur, Y., Benyagoub, S.: Realisability of choreographies. In: Herzig, A., Kontinen, J. (eds.) FoIKS 2020. LNCS, vol. 12012, pp. 263–280. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-39951-1_16
11. Farah, Z., Ait-Ameur, Y., Ouederni, M., Tari, K.: A correct-by-construction model for asynchronously communicating systems. Int. J. Softw. Tools Technol. Transf. **19**(4), 465–485 (2016). https://doi.org/10.1007/s10009-016-0421-6