Compositional Verification of Asynchronously Communicating Systems

Jan Martijn E.M. van der Werf(✉)

Department of Information and Computing Science, Utrecht University,
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands
j.m.e.m.vanderwerf@uu.nl

Abstract. Within a network of asynchronously communicating systems, the complete network is often not known, or even available at run-time. Consequently, verifying whether the network of communicating systems behaves correctly, i.e., the network does not contain any deadlock or livelock, is impracticable. As such systems are highly concurrent by nature, Petri nets form a natural choice to model these systems and their communication.

This paper presents a formal framework based on a generic communication condition to verify correctness of the system by pairwise checking whether these systems communicate correctly and fulfill some condition, then the whole network is guaranteed to behave correctly. As an example, this paper presents the elastic communication condition.

1 Introduction

Dividing the functionality of a system into subsystems such that each subsystem implements its own specific functionality is not new. Already in the sixties of the last century, McIlroy [17] suggested to use components to design and implement software systems. A component implements a specific part of the specification, masking its internal design [22].

A component offers some functionality, and, in order to deliver this, it uses functionality of other components. This way, a component has two roles: it is a *provider* and a *consumer*. From a business oriented view, a component *sells* functionality, and to meet its commitments, it *buys* functionality of other components [4,12].

With the advent of paradigms like Service Oriented Architectures [3,18], systems become more and more distributed. Some of the components of the system may be offered by third parties. As these third parties do not expose which components their systems use, the individual systems form a, possibly unknown, large scale ecosystem: a *dynamic network* of communicating components. These systems communicate via *messages*: a component *requests functionality* from another component, which in turn *eventually sends its answer*. Hence, communication between the components is *asynchronous* by nature. Verification of asynchronously communicating systems is known to be a hard problem.

The nature of this class of communicating systems is asymmetric. A provider commits itself to deliver some functionality. It does not matter what other

I. Lanese and E. Madelaine (Eds.): FACS 2014, LNCS 8997, pp. 49–67, 2015.
DOI: 10.1007/978-3-319-15317-9_4

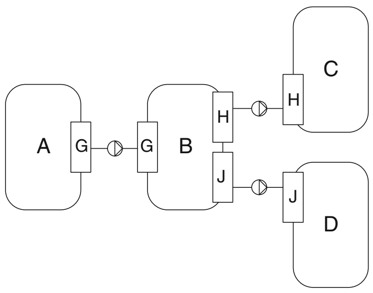

Fig. 1. Example of a component tree of four components A, B, C and D.

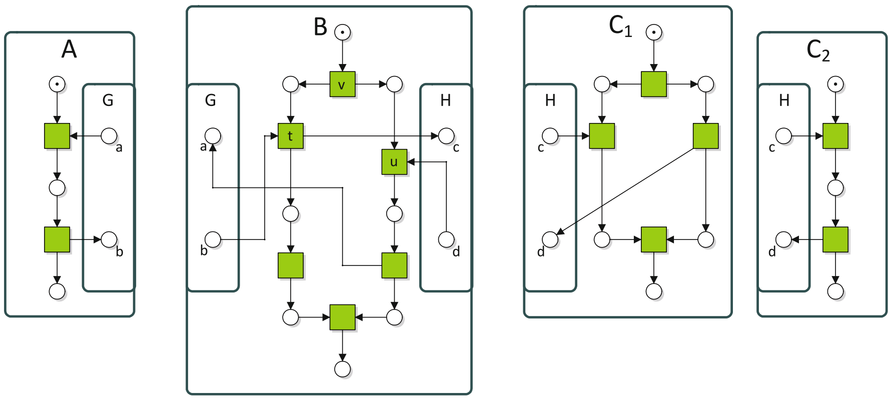

Fig. 2. Four components A, B, C_1 and C_2, such that $A \oplus_G B \oplus_H C_1$ is sound, but $A \oplus_G B \oplus_H C_2$ is not.

components that provider needs, as long as it keeps delivering the requested functionality. Therefore, the connections between components have a direction: they are initiated by some client, and accepted by a provider. Consider the component architecture depicted in Fig. 1. There are four components, A, B and C, which are connected via ports G, H and J. The $\rhd$ operator indicates the direction of the communication. In this example, component B delivers a service to component A over port G, and to do so, it uses the functionality of its children C and D.

At run-time, components use other components to deliver their functionality. In this way, the components form a *component tree*. The dynamic binding of components causes the component tree to be unknown at design time. This makes verification of behavioral correctness very hard. Thus, if we want to ensure behavioral correctness, we need a verification method that only considers pairwise compositions of components: if each component is sound, and all pairwise connected components satisfy some condition, the whole tree should be sound.

Current compositional verification techniques start with the verification of a pair of components, compose these into a larger component, and then use this larger composition for the next pairwise composition. However, this way of verification does not take the dynamic nature of service oriented approaches into account: if one of the components changes, the whole network needs to be checked again. Consider for example the components A, B, C_1 and C_2 depicted in Fig. 2. In this example, the composition of A, B and C_1 is sound. However, if B decides to use C_2 instead of C_1, the communication with A is hampered. Current compositional verification techniques have to re-verify the whole network for soundness again.

In [16], the authors prove that in general verification of such a dynamic, distributed setting is undecidable. Current research results (cf. [15,20,25,26]) are based on a message bound.

In this paper, we present a framework based on communication conditions to verify a subclass of asynchronously communicating systems compositionally [24]. The formal foundation of the framework is *Petri nets*, in which communication is asynchronous by nature. *Petri nets* can be used both for modeling the *internal activities* of a component, as well as for the interaction between components. We focus on *soundness* of systems: a system should always have a possibility to terminate.

This paper is structured as follows. Section 2 presents the basic notions used throughout the paper. Next, Sect. 3 introduces the notion of components and their composition. In Sect. 4, we present a general framework to verify correctness of component trees compositionally. Next, Sect. 5 shows a subclass of communicating systems based on this general framework. Section 6 concludes the paper.

2 Preliminaries

Let S be a set. The powerset of S is denoted by $\mathcal{P}(S) = \{S' \mid S' \subseteq S\}$. We use $|S|$ for the number of elements in S. Two sets U and V are *disjoint* if $U \cap V = \emptyset$. A *bag* m over S is a function $m : S \rightarrow \mathbb{N}$, where $\mathbb{N} = \{0, 1, 2, \ldots\}$ denotes the natural numbers. We denote e.g. the bag m with an element a occurring once, b occurring three times and c occurring twice by $m = [a, b^3, c^2]$. The set of all bags over S is denoted by $\mathbb{N}^S$. Sets can be seen as a special kind of bag were all elements occur only once. We use $+$ and $-$ for the sum and difference of two bags, and $=$, $<$, $>$, $\leq$, $\geq$ for the comparison of two bags, which are defined in a standard way. The projection of a bag $m \in \mathbb{N}^S$ on elements of a set $U \subseteq S$, is denoted by $m_{|U}$, and is defined by $m_{|U}(u) = m(u)$ for all $u \in U$ and $m_{|U}(u) = 0$ for all $u \in S \setminus U$. Furthermore, if for some $n \in \mathbb{N}$, disjoint sets $U_i \subseteq S$ with $1 \leq i \leq n$ exist such that $S = \bigcup_{i=1}^n U_i$, then $m = \sum_{i=1}^n m_{|U_i}$.

A *sequence* over S of length $n \in \mathbb{N}$ is a function $\sigma : \{1, \ldots, n\} \rightarrow S$. If $n > 0$ and $\sigma(i) = a_i$ for $i \in \{1, \ldots, n\}$, we write $\sigma = \langle a_1, \ldots, a_n \rangle$. The length of a sequence is denoted by $|\sigma|$. The sequence of length 0 is called the *empty sequence*, and is denoted by ϵ. The set of all finite sequences over S is denoted by S^*. We write $a \in \sigma$ if a $1 \leq i \leq |\sigma|$ exists such that $\sigma(i) = a$. *Concatenation* of two sequences $\nu, \gamma \in S^*$, denoted by $\sigma = \nu; \gamma$, is a sequence defined by

$\sigma : \{1, \ldots, |\nu| + |\gamma|\} \to S$, such that $\sigma(i) = \nu(i)$ for $1 \leq i \leq |\nu|$, and $\sigma(i) = \gamma(i - |\nu|)$ for $|\nu| + 1 \leq i \leq |\nu| + |\gamma|$.

A projection of a sequence $\sigma \in S^*$ on elements of a set $U \subseteq S$ (i.e. eliminating the elements from $S \setminus U$) is denoted as $\sigma_{|U}$. The bag denoted the elements of a sequence σ and their occurrences is called the *Parikh vector* and is denoted by $\overrightarrow{\sigma}$.

Labeled transition systems. To model the behavior of a system, we use a *labeled transition system.* A *labeled transition system* (LTS) is a 5-tuple $(S, \mathcal{A}, \to, s_0, \Omega)$ where S is a set of *states*; $\mathcal{A}$ is a set of *actions*; $\to \subseteq (S \times (\mathcal{A} \cup \{\tau\}) \times S)$ is a *transition relation*, where $\tau \notin \mathcal{A}$ is the silent action. $(S, \to, \emptyset)$ is a labeled directed graph, called the *reachability graph*; $s_0 \in S$ is the *initial state*; and $\Omega \subseteq S$ is the set of *accepting states.*

Let $L = (S, \mathcal{A}, \to, s_i, \Omega)$ be an LTS. For $s, s' \in S$ and $a \in \mathcal{A} \cup \{\tau\}$, we write $(L : s \xrightarrow{a} s')$ if and only if $(s, a, s') \in \to$. An action $a \in \mathcal{A} \cup \{\tau\}$ is called *enabled* in a state $s \in S$, denoted by $(L : s \xrightarrow{a})$ if a state s' exists such that $(L : s \xrightarrow{a} s')$. If $(L : s \xrightarrow{a} s')$, we say that state s' is *reachable* from s by an action labeled a. A state $s \in S$ is called a *deadlock* if no action $a \in \mathcal{A} \cup \{\tau\}$ exists such that $(L : s \xrightarrow{a})$. We define $\Longrightarrow$ as the smallest relation such that $(L : s \Longrightarrow s')$ if $s = s'$ or $\exists s'' \in S : (L : s \Longrightarrow s'' \xrightarrow{\tau} s')$. As a notational convention, we may write $\overset{\tau}{\Longrightarrow}$ for $\Longrightarrow$. For $a \in \mathcal{A}$, we define $\overset{a}{\Longrightarrow}$ as the smallest relation such that $(L : s \overset{a}{\Longrightarrow} s')$ if $\exists s_1, s_2 \in S : (L : s \Longrightarrow s_1 \xrightarrow{a} s_2 \Longrightarrow s')$.

We lift the notation of actions to sequences. For the empty sequence ϵ, we have $(L : s \xrightarrow{\epsilon} s')$ if and only if $(L : s \Longrightarrow s')$. Let $\sigma \in \mathcal{A}^*$ be a sequence of length $n > 0$, and let $s_0, s_n \in S$. Sequence σ is a *firing sequence*, denoted by $(L : s_0 \xrightarrow{\sigma} s_n)$, if states $s_{i-1}, s_i \in S$ exist such that $(L : s_{i-1} \overset{\sigma(i)}{\Longrightarrow} s_i)$ for all $1 \leq i \leq n$. We write $(L : s \xrightarrow{*} s')$ if a sequence $\sigma \in \mathcal{A}^*$ exists such that $(L : s \xrightarrow{\sigma} s')$, and say that s' is *reachable* from s. The set of reachable states from some state $s \in S$ is defined as $\mathcal{R}(L, s) = \{s' \mid (L : s \xrightarrow{*} s')\}$. We lift the notation of reachable states to sets by $\mathcal{R}(L, M) = \bigcup_{s \in M} \mathcal{R}(L, s)$ for $M \subseteq S$. A set of states $M \subseteq S$ is called a *livelock* if $M \subseteq \mathcal{R}(L, M)$. An LTS $L = (S, \mathcal{A}, \to, s_0, \Omega)$ is called *weakly terminating* if $\Omega \subseteq \mathcal{R}(L, s_0)$.

Petri nets. A *Petri net* [19] is a 3-tuple $N = (P, T, F)$ where (1) P and T are two disjoint sets of *places* and *transitions* respectively; (2) $F \subseteq (P \times T) \cup (T \times P)$ is a *flow relation.* The elements from the set $P \cup T$ are called the *nodes* of N. Elements of F are called *arcs.* Places are depicted as circles, transitions as squares. For each element $(n_1, n_2) \in F$, an arc is drawn from n_1 to n_2. Two Petri nets $N = (P, T, F)$ and $N' = (P', T', F')$ are *disjoint* if and only if $(P \cup T) \cap (P' \cup T') = \emptyset$. Let $N = (P, T, F)$ be a Petri net. Given a node $n \in (P \cup T)$, we define its *preset* ${}^\bullet_N n = \{n' \mid (n', n) \in F\}$, and its *postset* $n^\bullet_N = \{n' \mid (n, n') \in F\}$. We lift the notation of preset and postset to sets. Given a set $U \subseteq (P \cup T)$, ${}^\bullet_N U = \bigcup_{n \in U} {}^\bullet_N n$ and $U^\bullet_N = \bigcup_{n \in U} n^\bullet_N$. If the context is clear, we omit the N in the superscript.

A *marking* of N is a bag $m \in \mathbb{N}^P$, where $m(p)$ denotes the number of *tokens* in place $p \in P$. If $m(p) > 0$, place p is called *marked* in marking m. A Petri net

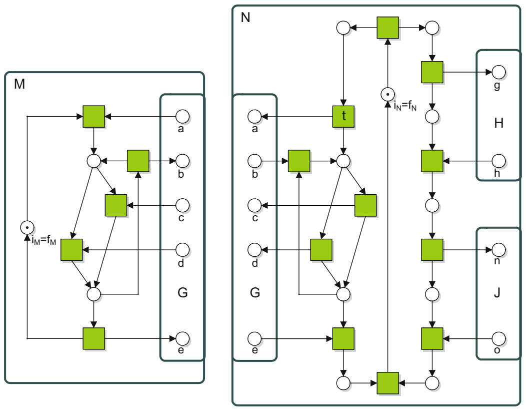

Fig. 3. Two components N and M with three ports G, H and J, where components N and M share port G.

N with corresponding marking m is written as (N, m) and is called a *marked Petri net.*

A *system* is a 3-tuple $S = (N, m_0, \Omega)$ where (N, m_0) is a marked Petri net with $N = (P, T, F)$ and $\Omega \subseteq \mathbb{N}^P$ is the set of *final markings*. Its semantics is defined by an LTS $\mathcal{N}(S) = (\mathbb{N}^P, T, \rightarrow, m_0, \Omega)$ such that $(m, t, m') \in\rightarrow$ iff ${}^\bullet t \leq m$ and $m' + {}^\bullet t = m + t^\bullet$ for $m, m' \in \mathbb{N}^P$ and $t \in T$. We write $(N : m \xrightarrow{t} m')$, $\mathcal{R}(N, m_0)$, $\mathcal{L}(N, m_0)$, and $\mathcal{T}(N, m_0)$ as a shorthand notation for $(\mathcal{N}(N, m_0) : m \xrightarrow{t} m')$, $\mathcal{R}(\mathcal{N}(N, m_0))$, $\mathcal{L}(\mathcal{N}(N, m_0))$, and $\mathcal{T}(\mathcal{N}(N, m_0))$, respectively. A marking $m \in \mathcal{R}(N, m_0)$ is a home marking if $m \in \mathcal{R}(N, m')$ for all $m' \in \mathcal{R}(N, m_0)$.

3 Asynchronously Communicating Systems

In a network of asynchronously communicating systems, systems communicate via message passing. We call these systems *components* of the network. Two components are connected via some interface that defines which messages are exchanged between the systems. As communication is asynchronous, Petri nets [19] form a natural choice to model the communication between these components. We model the different messages that can be sent and received via special places, called *interface places.* A component either receives messages from an interface place, which is then called an *input place*, or it sends messages to an interface place, which we then call an *output place.*

As components communicate with multiple components, we partition the interface places of a system into *ports.* A transition can send or receive messages via a port. For this, we introduce the notion of a transition sign. A transition sends messages to a port (sign !), receives messages from a port (sign ?) or does not communicate at all with a port (sign τ).

The marking of a component represents the internal state of the component, together with messages it has sent and received. As initially no messages have been sent or received, the initial marking of a component has no messages in its interface places. Similarly, in the desired final marking of a component, all messages have been processed, i.e., all interface places should be empty. Often, the desired final marking of a component represents an idle state, from which the component can respond on new messages again. In terms of Petri nets, the final marking is often a home marking.

Figure 3 depicts two components N and M. Component M has a single port G with three input places a, c and d, and two output ports b and e. Component N has three ports, G, H and J. The internal structure of a component, i.e., the component without the interface places, is called the *skeleton*.

Definition 1 (Component, skeleton, sign). *A* Component *is defined as an 8-tuple* $(P, I, O, T, F, \mathcal{G}, i, f)$ *where* $((P \cup I \cup O), T, F)$ *is a Petri net;* P *is a set of* internal places*;* I *is the set of* input places*,* O *is the set of* output places *such that* P*,* I *and* O *are pairwise disjoint and* ${}^\bullet I = O^\bullet = \emptyset$*;* $\mathcal{G} \subseteq \mathcal{P}(I \cup O)$ *is a partitioning of the interface places, an element of* $\mathcal{G}$ *is called a* port*; a transition either sends or receives messages, i.e.,* ${}^\bullet G \cap G^\bullet = \emptyset$ *for all* $G \in \mathcal{G}$*.* $i \in \mathbb{N}^P$ *is the initial marking, and* $f \in \mathbb{N}^P$ *is the final marking.*

Two components N *and* M *are called* disjoint *if* $(P_N \cup I_N \cup O_N \cup T_N) \cap (P_M \cup I_M \cup O_M \cup T_M) = \emptyset$*. A component* N *is called* closed *if* $I_N = O_N = \emptyset$*. The set of all components is denoted by* $\mathfrak{N}$*. As a shorthand notation, we write* $\mathcal{R}(N, m)$ *for* $\mathcal{R}((P_N \cup I_N \cup O_N, T_N, F_N), m)$ *for* $m \in \mathbb{N}^{P_N \cup I_N \cup O_N}$*.*

The skeleton *of* N *is defined as the Petri net* $\mathcal{S}(N) = (P_N, T_N, F)$ *with* $F = F_N \cap ((P_N \times T_N) \cup (T_N \times P_N))$*. The* skeleton system *of* N *is defined as the system* $\overline{\mathcal{S}}(N) = (\mathcal{S}(N), i_N, \{f_N\})$*.*

The sign *of a transition with respect to a port* $G \in \mathcal{G}$ *is a function* $\lambda_G : T \to \{!, ?, \tau\}$ *defined by* $\lambda_G(t) = !$ *if* $t^\bullet \cap G \neq \emptyset$*,* $\lambda_G(t) = ?$ *if* ${}^\bullet t \cap G \neq \emptyset$*, and* $\lambda_G(t) = \tau$ *otherwise, for all* $t \in T$*.*

It is desired that from every reachable marking of a component, the component should be able to reach its desired final marking. This property is expressed in the notion of weak termination. Another basic sanity check for components is to check whether it internally behaves correctly, i.e., ignoring the interface places, the component should be able to always reach its final marking. As this property is closely related to soundness of workflow nets [1], We call this property *soundness.*

Definition 2 (Weak termination and soundness). *Let* N *be a component. It is* weakly terminating*, if for each marking* $m \in \mathcal{R}(N, i_N)$*, we have* $f_N \in \mathcal{R}(N, m)$*. It is* sound*, if the system defined by its skeleton is weakly terminating.*

Notice that this definition does not require the final marking of a component to be a deadlock. Instead, the final marking can be seen as a home marking, in which the component is in rest.

Components communicate via their ports. To be able to compose two components so that they are able to communicate, the components should have inverted ports: input places of the one should be output places of the other, and vice versa.

Definition 3 (Composition of components). *Two components A and B are* composable *with respect to port $G \in \mathcal{G}_A \cap \mathcal{G}_B$, denoted by $A \oplus_G B$, if and only if $(P_A \cup I_A \cup O_A \cup T_A) \cap (P_B \cup I_B \cup O_B \cup T_B) = (I_A \cap O_B) \cup (O_A \cap I_B) = G$.*

If A and B are composable with respect to port G, their composition *results in a component $A \oplus_G B = (P, I, O, T, F, \mathcal{G}, i, f)$ where $P = P_A \cup P_B \cup H$; $I = (I_A \cup I_B) \setminus H$; $O = (O_A \cup O_B) \setminus H$; $T = T_A \cup T_B$; $F = F_A \cup F_B$; $\mathcal{G} = (\mathcal{G}_A \cup \mathcal{G}_B) \setminus \mathcal{H}$; $i = i_A + i_B$; and $f = f_A + f_B$. If a port $G \in \mathcal{G}_A \cap \mathcal{G}_B$ exists such that $A \oplus_G B$, we write $A \oplus B$.*

Consider again the components N and M of Fig. 3. Both components share port G, where the input places a, c and d of M are output places of N, and the output places b and d are input places of N. Their composition results in a component $N \oplus M$, where the places a, b, c, d and e become internal places of the composition.

The composition operator is commutative and associative, provided that the components are composable.

Corollary 4 (Composition is commutative and associative). *Let A, B and C be three components, such that $A \cap C = \emptyset$, and let $G \in \mathcal{G}_A \cap \mathcal{G}_B$ and $H \in \mathcal{G}_B \cap \mathcal{G}_C$. If A and B are composable w.r.t some port $G \in \mathcal{G}_A \cap \mathcal{G}_B$, then $A \oplus_G B = B \oplus_G A$; Also, $(A \oplus_G B) \oplus_H C$ exists iff $A \oplus_G (B \oplus_H C)$ exists. If the compositions exist, they are identical.*

In the remainder of this section, we discuss some properties of the composition operator. Composition only restricts behavior, i.e., the composition of two components A and B does not introduce any new behavior. In [24], it is shown that the projection of a composition to either one of its constituents is a simulation relation [10]. As a consequence, a firing sequence in the composition of two components is a firing sequence of its constituents, after hiding the transitions of the other component, and any reachable marking in the composition results in a reachable marking of that constituent.

Corollary 5. *Let A and B be two composable components with respect to some port $G \in \mathcal{G}_A \cap \mathcal{G}_B$. Define $N = A \oplus_G B$. Let $m, m' \in \mathcal{R}(\overline{\mathcal{S}}(N))$ and $\sigma \in T_N^*$ such that $(\mathcal{S}(N) : m \xrightarrow{\sigma} m')$. Then $m_{|P_A} \in \mathcal{R}(\overline{\mathcal{S}}(A))$ and $m_{|P_B} \in \mathcal{R}(\overline{\mathcal{S}}(B))$, $(\mathcal{S}(A) : m_{|P_A} \xrightarrow{\sigma_{|T_A}} m'_{|P_A})$, and $(\mathcal{S}(B) : m_{|P_B} \xrightarrow{\sigma_{|T_B}} m'_{|P_B})$.*

4 A General Verification Framework

In this section we present a formal framework for compositional verification of soundness on component trees. Proving the soundness of a component tree is

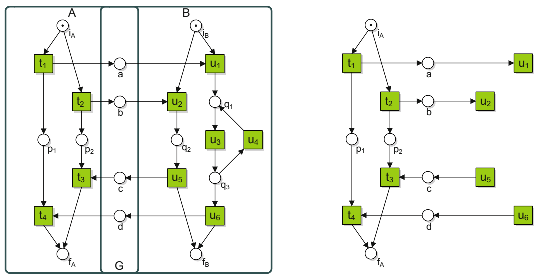

Fig. 4. Composition $A \oplus_G B$ and its subnet $N_1 = \mathcal{C}_B(A)$

done in two steps. First of all, each component should be sound itself. Next, each connection is checked against some communication condition, from which soundness of the composition, and of the whole tree can be concluded. Such a condition should satisfy some criteria. A component may not notice the difference whether it is communicating with a single component or with a component tree. We therefore search for a *sequence relation* $\varphi : T_N^* \times T_N^* \to \mathbb{B}$, which is a predicate on the firing sequences of component N, such that this property is guaranteed.

As shown in [4], soundness is not a sufficient condition. Consider for example the composition in Fig. 2. In this example, it is easy to verify that both compositions $A \oplus_G B$ and $B \oplus_H C_2$ are sound. However, in the composition $A \oplus_G B \oplus_H C_2$, transition t is only enabled once it received a message from component A, which in turn requires a message from component C_2. Consequently, the composition of the tree is not sound.

As soundness is not a sufficient condition, we need to strengthen the soundness property by stating that for all reachable markings in the composition of B and C and firing sequence σ in B, a firing sequence $\tilde{\sigma}$ should exist in the composition such that σ and $\tilde{\sigma}$ satisfy the predicate φ.

Definition 6 (Communication condition). *Let B and C be two components such that $B \oplus_H C$ for some $H \in \mathcal{G}_B \cap \mathcal{G}_C$. Define $N = B \oplus_H C$ and let $\varphi : T_B^* \times T_N^* \to \mathbb{B}$ be a sequence relation. The* communication condition $\mathrm{com}_\varphi(B, C)$ *holds if and only if:*

$$\forall m \in \mathcal{R}(\mathcal{S}(N), i_N), \sigma \in T_B^* :$$
$$(\mathcal{S}(B) : m_{|P_B} \xrightarrow{\sigma} f_B) \implies (\exists \tilde{\sigma} \in T_N^* : (\mathcal{S}(N) : m \xrightarrow{\tilde{\sigma}} f_N) \wedge \varphi(\sigma, \tilde{\sigma}))$$

In fact, the communication condition states that $B \oplus C$ is able to follow B. For any φ, this condition implies soundness, which directly follows from Corollary 5.

Lemma 7 (φ-communication condition implies soundness). *Let B and C be two components that are composable with respect to port $G \in \mathcal{G}_B \setminus \mathcal{G}_C$. Let φ be a sequence relation. If B is sound and* $\mathrm{com}_\varphi(B, C)$ *holds for some sequence relation φ, then $B \oplus_H C$ is sound.*

Condition com_φ is sufficient for deciding the soundness of two components. Let A, B and C be three components such that A communicates with B, B communicates with C, but A and C do not communicate, i.e., A and C are disjoint. We prove that if the composition $A \oplus B$ is sound, and components B and C satisfy $\mathrm{com}_\varphi(B, C)$, then the composition of A, B and C is sound. In order to provide a sufficient condition for concluding soundness of a tree of three components, such a sequence relation needs to satisfy several criteria. These criteria follow directly from the proof.

To prove soundness of the component tree, we need to show that given a reachable marking of the component tree, the final marking should be reachable. As the composition of A and B is sound, we have a firing sequence in $A \oplus B$ from this marking leading to the final marking of $A \oplus B$. Condition $\mathrm{com}_\varphi(B, C)$ should guarantee that this firing sequence projected on B is still possible in the component tree. The condition ensures the existence of a firing sequence in $B \oplus C$ such that it satisfies the sequence relation φ.

Hence, we have a firing sequence in $A \oplus B$ and a firing sequence in $B \oplus C$ satisfying the sequence relation φ. We should be able to interweave these firing sequences, so that the resulting sequence is a firing sequence in the component tree. Therefore, we divide the composition of $A \oplus B$ into two subnets, N_1 and N_2. The first subnet, N_1, covers component A and the transitions of B that communicate with A. Figure 4 depicts the division of the composition $A \oplus B$ into N_1. Net N_2 is the skeleton of component B. Note that the union of nets N_1 and N_2 is the skeleton of the composition. The transitions of B that communicate with A are common for the two subnets, the places of N_1 and N_2 are disjoint.

Definition 8. *Let A and B be two components such that A and B are composable with respect to some port $G \in \mathcal{G}_A \cap \mathcal{G}_B$. Define $N = A \oplus_G B$. The Petri net $\mathcal{C}_B(A)$ is defined as $\mathcal{C}_B(A) = (P, T, F)$ where $P = P_A \cup G$, $T = T_A \cup {}_N^{\bullet}G \cup G_N^{\bullet}$ and $F = F_N \cap ((P \times T) \cup (T \times P))$.*

Every firing sequence in $A \oplus B$ can be turned into a firing sequence of $\mathcal{C}_B(A)$ by leaving out all transitions of T_B, except the transitions of B that communicate with A. The proof follows directly from Corollary 5.

Corollary 9. *Let A and B be two OPNs that are composable with respect to some port $G \in \mathcal{G}_A \cap \mathcal{G}_B$. Define $N = A \oplus B$ and $L = \mathcal{C}_B(A)$. Then for all $\sigma \in T_N^*$ and $m, m' \in \mathbb{N}^{P_N}$ such that $(\mathcal{S}(N) : m \xrightarrow{\sigma} m')$ holds $(L : m_{|P_L} \xrightarrow{\sigma_{|T_L}} m'_{|P_L})$.*

In the soundness proof, the firing sequence in $A \oplus B$ is projected on $\mathcal{C}_B(A)$, and it will be interweaved with the resulting firing sequence of the communication condition. The interweaving property will guarantee that this interweaving is possible.

Property 10 (Interweaving firing sequences). Let A and B be two components that are composable with respect to some port $G \in \mathcal{G}_A \cap \mathcal{G}_B$. Let φ be a sequence relation as defined in Definition 6. Let $N_1 = \mathcal{C}_B(A)$ and $N_2 = \mathcal{S}(B)$. Let $\mu \in T_{N_1}^*$ and $m, m' \in \mathbb{N}^{P_{N_1}}$ such that $(N_1 : m \xrightarrow{\mu} m')$. Let $\nu \in T_{N_2}^*$ and $\overline{m}, \overline{m}' \in \mathbb{N}^{P_{N_2}}$

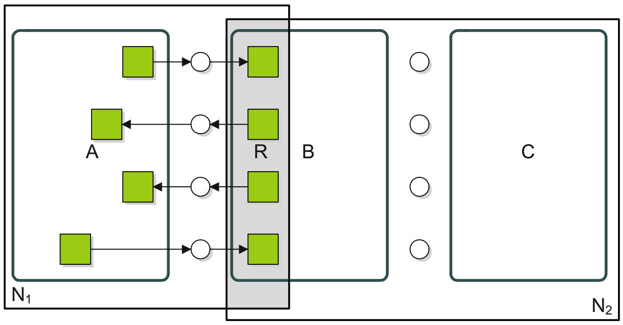

Fig. 5. The composition $A \oplus B \oplus C$ is split into $N_1 = \mathcal{C}_B(A)$ and $N_2 = \mathcal{S}(B \oplus C)$

such that $(N_2 : \overline{m} \xrightarrow{\nu} \overline{m}')$ and $\varphi(\mu, \nu)$. Then a $\sigma \in T_N^*$ exists such that $(\mathcal{S}(N) : m + \overline{m} \xrightarrow{\sigma} m' + \overline{m}')$, $\varphi(\mu, \sigma)$ and $\varphi(\nu, \sigma)$.

The interweaving property expresses that two sequences can be combined into a single firing sequence that is executable and satisfies the sequence relation. Also, the sequence relation should hold for a firing sequence, and its firing sequence in which all transitions are hidden except for the transitions that communicate. Rephrased, the sequence relation φ should not consider all transitions in B, but only the transitions of B that communicate with A. This is expressed in the next property.

Property 11. Let B and C be two components that are composable, and let $G \in \mathcal{G}_B \setminus \mathcal{G}_C$. Define $N = B \oplus C$ and $R = {}_N^{\bullet}G \cup G_N^{\bullet}$. Let φ be a sequence relation as defined in Definition 6. Let $\sigma \in T_B^*$ and $\tilde{\sigma} \in T_N^*$. If $\varphi(\sigma, \tilde{\sigma})$, then $\varphi(\sigma_{|R}, \tilde{\sigma})$ and $\varphi(\sigma, \tilde{\sigma}_{|R})$.

This leads to the main theorem, that the communication condition com_φ is a sufficient condition for soundness. Note that to prove the main theorem for a specific sequence relation, we need to show that both properties hold for the sequence relation.

Theorem 12 (Communication condition sufficient for soundness). *Let A, B and C be three components such that A and B are composable with respect to port $G \in \mathcal{G}_A \cap \mathcal{G}_B$, B and C are composable, A and C are disjoint and $A \oplus_G B$ is sound. Let φ be a sequence relation as defined in Definition 6.*

If $\mathrm{com}_\varphi(B, C)$ holds, then $A \oplus_G B \oplus_H C$ is sound.

Proof. Define $N = A \oplus_G B \oplus_H C$, $M = A \oplus_G B$, $N_2 = \mathcal{S}(B \oplus_H C)$ and $N_1 = \mathcal{C}_B(A)$. Let $m \in \mathcal{R}(\mathcal{S}(N))$. Since M is sound, a $\sigma \in T_M^*$ exists such that $(\mathcal{S}(M) : m_{|P_M} \xrightarrow{\sigma} f_M)$. By Corollary 5, the firing sequence $\sigma_{|T_B}$ is also a firing sequence in $\mathcal{S}(B)$, i.e. $(\mathcal{S}(B) : m_{|P_B} \xrightarrow{\sigma_{|T_B}} f_B)$. By Corollary 5, $m_{|P_{N_2}} \in \mathcal{R}(N_2, i_M)$. Hence, we can apply the communication condition $\mathrm{com}_\varphi(B, C)$ on $m_{|P_{N_2}}$ and $\sigma_{|T_B}$, which results in a firing sequence $\tilde{\sigma} \in T_{N_2}^*$ such that $(N_2 : m_{|P_{N_2}} \xrightarrow{\tilde{\sigma}} f_{N_2})$ and

$\varphi(\sigma_{|T_B}, \tilde{\sigma})$. Hence, we have a firing sequence σ in M and a firing sequence $\tilde{\sigma}$ in N_2, which we need to interweave.

We split the composition N in N_1 and N_2, as shown in Fig. 5. By Corollary 9, $(N_1 : m_{|P_{N_1}} \xrightarrow{\sigma_{|T_{N_1}}} f_A)$. By Property 11, the sequence relation also holds for the projected firing sequence, i.e. $\varphi(\sigma_{|T_{N_1}}, \tilde{\sigma})$ holds. Then the Interweaving Property (10) applied on $(N_1, m_{|P_{N_1}})$ with firing sequence $\sigma_{|T_{N_1}}$ and $(N_2, m_{|P_{N_2}})$ with firing sequence $\tilde{\sigma}$ results in a firing sequence $\overline{\sigma} \in T_N^*$ such that $(\mathcal{S}(N) : m \xrightarrow{\overline{\sigma}} f_A + f_{N_2} = f_N)$. Hence, N is sound. $\square$

From Theorem 12, it follows that com_φ is a sufficient condition to conclude soundness of a component tree consisting of three components if Properties 10 and 11 hold for the sequence relation. Hence, if two connected components satisfy com_φ, the composition is guaranteed to be sound, and it can be used for compositional verification. In fact, $\text{com}_\varphi(A, B)$ implies a direction in the component tree: component A *uses* component B to provide its service on port G, or, rephrased, B *provides a service* to A.

Definition 13 (Component uses another component). *Let A and B be two composable components with respect to port $G \in \mathcal{G}_A \cap \mathcal{G}_B$, and let φ be a sequence relation as defined in Definition 6.*

We say A uses *B, denoted by $A \circledR_\varphi B$, if $A \oplus_G B$ and $\text{com}_\varphi(A, B)$.*

In this way, we can construct a *component tree* of components that uses other components to deliver their service. A component tree is a tree of components connected to each other such that components can only "subcontract" work to other components. The structure of the tree is defined by the tree function c. Each node A is a component that delivers a service to its parent $c(A)$ using the services of its children $c^{-1}(A)$. Each component only communicates with its parent and its children, communication with other components is not allowed. Note that the communication implied by this function is asymmetric: the parent uses its children to deliver the service requested. By requiring that the transitive closure of c is irreflexive, we ensure the component tree to be a tree.

Definition 14 (component tree). *A* component tree *is a pair $(\mathcal{O}, c)$ where $\mathcal{O}$ is a set of components, and $c : \mathcal{O} \rightharpoonup \mathcal{O}$ is a partial function called the* parent function *such that the transitive closure c^* of c is irreflexive, for all $A, B \in \mathcal{O}$:*

- *$c(B) = A \implies |\mathcal{G}_A \cap \mathcal{G}_B| = 1 \wedge A \circledR_\varphi B$; and*
- *$A \cap B \neq \emptyset \implies c(A) = B \vee c(B) = A$.*

and for all $A \in \mathcal{O}$ a $B \in \mathcal{O}$ exists such that $(A, B) \in c^*$ or $(B, A) \in c^*$.

An example is shown in Fig. 1, where component A uses component B, which in turns uses components C and D.

In a component tree, each parent should use the services of its children. Hence, if the root is sound, and each parent uses its children, the component tree should be sound. This is expressed in the next theorem. The proof uses the associativity and commutativity of the composition operator and Theorem 12.

Theorem 15 (Soundness of component trees). *Let $(\mathcal{O}, c)$ be a component tree. If all components of $\mathcal{O}$ are sound, then $\bigoplus_{X \in \mathcal{O}} X$ is sound.*

Proof. Assume all components in $\mathcal{O}$ are sound. As $(\mathcal{O}, c)$ is a tree, a topological sort $\sqsubseteq$ exists on the nodes $\mathcal{O}$. Let $\mathcal{O} = \{O_1, \ldots, O_n\}$ such that $O_i \sqsubseteq O_{i+1}$ for $1 \leq i < n$. We prove the lemma by induction on i. For $i = 1$, the statement holds trivially.

Now assume $1 < i < n$ and $\bigoplus_{X \in \mathcal{O}'} X$ is sound where $\mathcal{O}' = \{O_1, \ldots, O_i\}$. Let $B = O_{i+1}$. Since $\sqsubseteq$ is a topological sort, there exists a unique $A \in \mathcal{O}'$ such that $A \circledtriangleright_{\varphi} B$, and B is disjoint with all OPNs in $\mathcal{O}' \setminus \{A\}$.

By associativity and commutativity, we have $\bigoplus_{X \in \mathcal{O}'} X = (\bigoplus_{X \in \mathcal{O}' \setminus \{A\}} X) \oplus A$, and $\bigoplus_{X \in \mathcal{O}' \setminus \{A\}} X$ is disjoint with B. As $A \circledtriangleright_{\varphi} B$, we have $\mathrm{com}_{\varphi}(A, B)$, and thus by Theorem 12, $(\bigoplus_{X \in \mathcal{O}' \setminus \{A\}} X) \oplus A \oplus_G B$ is sound. Again by associativity and commutativity, $(\bigoplus_{X \in \mathcal{O}' \setminus \{A\}} X) \oplus A \oplus_G B = \bigoplus_{X \in \mathcal{O}' \cup \{B\}} X$. Hence, the statement holds. □

5 Elastic Communication

In this section, we present a communication condition that satisfies both Properties 10 and 11. Let A, B and C be three components such that A and B, and B and C are both composable, and A and C are disjoint. In [4], it is shown that checking whether the composition $B \oplus C$ behaves as component B on the interface with A, i.e., identical communication, is sufficient to prove soundness of a component tree. In fact, it is easy to show that this condition satisfies Properties 10 and 11 [24]. However, this identical communication condition is very restrictive. One way to weaken the condition of [4] is by allowing to permute port transitions within a communication block, i.e., a block of only sending or receiving transitions, possibly interweaved with silent transitions. Although this already weakens the condition, it remains very restrictive [24]. Consider for example the composition of Fig. 6. It is clear that the composition $A \oplus_G B$ is sound. Now, take the sequence $\sigma = \langle t_1, t_2, t_3, t_4, t_5 \rangle$. Although it is easy to verify that the composition $A \oplus B \oplus C$ is sound, no firing sequence can be found that behaves as σ, even when swaps within the same communication block is allowed. The main problem of the net is that the b message is sent too early for some of the sequences. This example shows that messages may be sent earlier without violating the soundness property. Soundness only requires that messages should be on time, i.e., components may send messages earlier, as long as they can both terminate properly. We reflect this in the *elastic communication condition*.

The condition allows sending transitions to be shuffled, as long as for each receiving transition at least the same sending transitions occur, or rephrased, sending transitions may occur at any position within its communication block, or it can be moved forward in the firing sequence. Although transitions sending messages may be moved forward in the firing sequence, the condition ensures that from every marking reachable, the final marking is reachable.

Consider the composition $A \oplus_G B$ of two components A and B. In the proof of Theorem 12, the composition is split into two nets, $N_1 = \mathcal{C}_B(A)$ and $N_2 = \mathcal{S}(B)$,

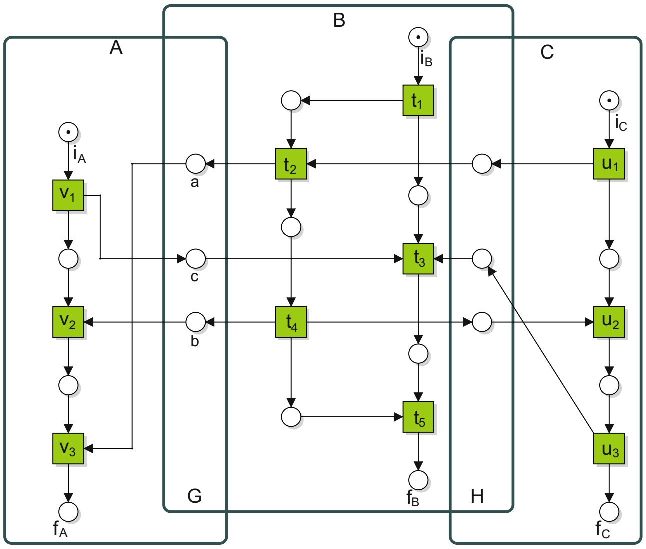

Fig. 6. Although net $A \oplus_G B \oplus_H C$ is sound, identical communication does not hold

and a firing sequence in N_1 is interweaved with a firing sequence in N_2. Let μ be a firing sequence in N_1 and ν a firing sequence in N_2. To be able to interweave the two firing sequences, ν has to produce the tokens it sends in time, and μ has to ensure that ν has sufficient tokens to be able to produce these tokens. In net N_1, all transitions of B either have an empty preset, or an empty postset. The set of transitions of B with an empty preset is labeled R_{out}, the set of transitions of B with an empty postset is labeled R_{in}. If in μ a transition of R_{out} fires, it means that a message from B is needed for A to continue. On the other hand, firing a transition of R_{in} in ν indicates that B needs a message from A.

To interweave sequences μ and ν into a firing sequence σ in the composition, if a transition in R_{in} is the next transition of ν to be added to σ, then the transition should already have fired in μ, since otherwise the transition cannot be enabled in the composition. Likewise, if a transition in R_{out} is the next transition of μ to be added to σ, then the transition should already have fired in ν, since otherwise the transition cannot be enabled in σ. If both conditions do not hold, we cannot create a firing sequence in the composition. Hence, the following formula has to hold:

$$\neg\exists\ 0 \le k < |\mu|, 0 \le l < |\nu| : \\ (\overrightarrow{\mu_{[1..k+1]}}_{|R_{out}} > \overrightarrow{\nu_{[1..l]}}_{|R_{out}}) \ \wedge\ (\overrightarrow{\nu_{[1..l+1]}}_{|R_{in}} > \overrightarrow{\mu_{[1..k]}}_{|R_{in}})$$

If such a pair k, l would exist, we cannot interweave the firing sequences: we cannot add the next transition of μ, since it needs tokens of ν that are not yet generated, and we cannot add the next transition of ν, since that transition needs tokens of μ that are not yet generated. If such a pair does not exist, we say the sequences are *elastic* to each other.

Definition 16 (Elastic sequences). *Let $N = (P, T, F)$ be a Petri net and $G \subseteq P$. Define $R_{in} = \{t \in T \mid \lambda_G(t) = ?\}$ and $R_{out} = \{t \in T \mid \lambda_G(t) = !\}$. Let $\mu, \nu \in T_N^*$. Sequence μ* is elastic to *sequence ν, denoted by $\mu \rightarrowtail_G \nu$ if and only if: $(\overrightarrow{\mu_{[1..k+1]}}_{|R_{out}} \leq \overrightarrow{\nu_{[1..l]}}_{|R_{out}}) \vee (\overrightarrow{\nu_{[1..l+1]}}_{|R_{in}} \leq \overrightarrow{\mu_{[1..k]}}_{|R_{in}})$ for all $0 \leq k < |\mu|$ and $0 \leq l < |\nu|$.*

Consider again Fig. 6. As an example, take the firing sequences $\sigma = \langle t_1, t_2, t_3, t_4, t_5\rangle$ and $\tilde{\sigma} = \langle t_1, u_1, t_2, t_4, u_2, u_3, t_3, t_5\rangle$. In $\tilde{\sigma}$, the firing of transition t_3 is moved forward with respect to σ, i.e., sending message b is "delayed" in σ. Since $\sigma_{[1..0]} = \epsilon$ for each firing sequence σ, we have by definition $\sigma_{[1..0]} \rightarrowtail_G \tilde{\sigma}_{[1..0]}$. The index of $\tilde{\sigma}$ may be increased up to the situation that $\sigma_{[1..0]} \rightarrowtail_G \tilde{\sigma}_{[1..6]}$, since then t_3 needs to be fired in $\sigma_{[1..0]} = \epsilon$, which is obviously not the case. Hence, we need to increase the index of σ, which is allowed up to $\sigma_{[1..5]} \rightarrowtail_G \tilde{\sigma}_{[1..6]}$. Then, it is allowed to increase the index of $\tilde{\sigma}$ up to $\sigma_{[1..5]} \rightarrowtail_G \tilde{\sigma}_{[1..8]}$. Hence, σ is elastic to $\tilde{\sigma}$. The sequences not only should be elastic, but also the number of messages sent and received by both sequences should match. These two requirements form the elastic sequence relation.

Definition 17 (Elastic communication condition). *Let B and C be two components that are composable with respect to port $G \in \mathcal{G}_B \setminus \mathcal{G}_C$. Let $\mu \in T_B^*$ and $\nu \in T_{B \oplus_H C}^*$. We define the* elastic sequence relation *$\psi_G : T_B^* \times T_{B \oplus_H C}^* \to \mathbb{B}$ by $\psi_G(\mu, \nu)$ if and only if $\overrightarrow{\mu}_{|R} = \overrightarrow{\nu}_{|R}$ and $\mu_{|R} \rightarrowtail_G \nu$, where $R = \{t \in T_B \mid \lambda_G(t) \neq \tau\}$. The* elastic communication condition *is defined as* $\text{com}_{\psi_G}(B, C)$.

In order to show that the elastic communication condition com_ψ is a sufficient condition, we need to show that Properties 10 and 11 hold for the elastic sequence relation. The latter follows directly from the definition of the elastic sequence relation.

Corollary 18. *Let A and B be two components that are composable with respect to port $G \in \mathcal{G}_B \setminus \mathcal{G}_C$. Define $N = A \oplus B$. Let $\mu \in T_B^*$ and $\nu \in T_N^*$ such that $\psi_G(\mu, \nu)$. Define $R = {}_N^{\bullet}G \cup G_N^{\bullet}$. Then $\psi_G(\mu_{|R}, \nu)$ and $\psi_G(\mu, \nu_{|R})$.*

To combine a firing sequence μ with a firing sequence ν it is elastic to, we need to consider the elasticity, i.e., the structure of the sequences. Hence, to prove Property 10 for ψ, we need to show that we can interweave firing sequences μ and ν. If $\overrightarrow{\mu_{[1..k+1]}}_{|R_{out}} \leq \overrightarrow{\nu_{[1..l]}}_{|R_{out}}$, we concatenate σ and $\langle \mu(k+1)\rangle$ if $\mu(k+1)$ is not in R_{in} or R_{out}, and if $\overrightarrow{\nu_{[1..l+1]}}_{|R_{in}} \leq \overrightarrow{\mu_{[1..k]}}_{|R_{in}}$, we concatenate σ and $\langle \nu(l+1)\rangle$. Since μ is elastic to ν, always at least one of the two cases holds for each $k < |\mu|$ and $l < |\nu|$. This operation results in the algorithm `IsElasticTo`. In the algorithm, the If-$\Box$-Fi construction indicates that if multiple guards are true, non-deterministically one of the guards evaluating true is chosen.

In this algorithm, if both conditions of the if clauses fail, sequence μ cannot be elastic to sequence ν, and hence, the algorithm fails. Otherwise, an interweaved firing sequence σ is returned, such that both $\mu \rightarrowtail_G \sigma$ and $\nu \rightarrowtail_G \sigma$.

Procedure IsElasticTo(μ,ν)

```
(k, l, σ) :=(0, 0, ε);
{Inv: $\mu_{[1..k]} \rightarrowtail_G \nu_{[1..l]} \wedge \mu_{[1..k]} \rightarrowtail_G \sigma \ \wedge \ \nu_{[1..l]} \rightarrowtail_G \sigma$ }
while $(k < |\mu| \vee l < |\nu|)$ do
    if $k < |\mu| \wedge \overrightarrow{\mu_{[1..k+1]}}_{|R_{out}} \leq \overrightarrow{\nu_{[1..l]}}_{|R_{out}}$ then
        if $\mu(k+1) \notin (R_{in} \cup R_{out})$ then
            $\sigma := \sigma; \langle \mu(k+1) \rangle$;
        fi
        $k := k+1$;
    □ $l < |\nu| \wedge \overrightarrow{\nu_{[1..l+1]}}_{|R_{in}} \leq \overrightarrow{\mu_{[1..k]}}_{|R_{in}}$ then
        $(\sigma, l) :=(\ \sigma; \langle \nu(l+1) \rangle, l+1)$;
    else
        return $\epsilon$;
    fi
od
return $\sigma$
```

Corollary 19. *Let N be a component and let $G \in \mathcal{G}_N$. Let $\mu, \nu \in T_N^*$. Then an invariant for procedure* `isElasticTo`*(μ,ν) is*

$$\mu_{[1..k]} \rightarrowtail_G \nu_{[1..l]} \ \wedge \ \mu_{[1..k]} \rightarrowtail_G \sigma \ \wedge \ \nu_{[1..l]} \rightarrowtail_G \sigma$$

Next, we need to show that the firing sequence constructed via `IsElasticTo` is executable. Given two OPNs A and B that are composable with respect to port G, we split the composition into $N_1 = \mathcal{C}_B(A)$ and $N_2 = \mathcal{S}(B)$. Every marking in the composition can be split into a marking in $\mathcal{S}(A)$, $\mathcal{S}(B)$ and some tokens in the interface places G. The marking in the interface G can again be split into places that are input for B, which we name x, and places that are output for B, which we name y. As shown in the next lemma, the elastic communication condition ensures that at each point in time, there are sufficient tokens in the interface places to continue.

Lemma 20. *Let A and B be two components such that they are composable with respect to some port $G \in \mathcal{G}_A \cap \mathcal{G}_B$. Define $G_I = G \cap I_B$, $G_O = G \cap O_B$, $N_1 = \mathcal{C}_B(A)$, $N_2 = \mathcal{S}(B)$ and $N = N_1 \cup N_2$. Let $m_0 \in I\!N^{P_{N_1}}$ be a marking, and let $\mu \in T_{N_1}^*$ be a firing sequence of length k such that for all $1 \leq i \leq |\mu|$, markings $m_{i-1}, m_i \in I\!N^{P_{N_1}}$ exist with $(N_1 : m_{i-1} \xrightarrow{\mu(i)} m_i)$. Let $\overline{m}_0 \in I\!N^{P_B}$ be a marking, and let $\nu \in T_B^*$ be a firing sequence of length l such that $\mu \rightarrowtail_G \nu$ and for all $1 \leq i \leq |\nu|$, markings $\overline{m}_{i-1}, \overline{m}_i \in I\!N^{P_{N_2}}$ exist with $(N_2 : \overline{m}_{i-1} \xrightarrow{\nu(i)} \overline{m}_i)$. Then, a firing sequence $\sigma \in T_N^*$ and a marking $m \in I\!N^{P_N}$ exist such that: (1) $\sigma =$* `IsElasticTo`*(μ, ν); (2) $\sigma_{|T_A} = \mu_{|T_A}$ and $\sigma_{|T_B} = \nu_{|T_B}$; (3) $(N : m_0 + \overline{m}_0 \xrightarrow{\sigma} m)$; and (4) $m_{k|P_A} \leq m$, and $\overline{m}_l \leq m$.*

Proof. Define $R_{in} = \{t \in T_B \mid \lambda_G(t) =?\}$, $R_{out} = \{t \in T_B \mid \lambda_G(t) =!\}$ and $R = R_{in} \cup R_{out}$. Note that $R = T_{N_1} \setminus T_{N_2}$.

We prove the lemma by induction on the structure of $\mu \rightarrowtail_G \nu$. The statement holds trivially for $\sigma = \epsilon$ and $m = m_0 + \overline{m}_0$.

Suppose the statement holds for some $\mu' \leq \mu$ and $\nu' \leq \nu$ such that $\mu' \rightarrowtail_G \nu'$, i.e. let $k = |\mu'|$ and $l = |\nu'|$, then for μ' and ν' a firing sequence $\sigma' \in T_N^*$ and marking $m' \in I\!N^{P_N}$ exist such that $\sigma' = \texttt{IsElasticTo}(\mu', \nu')$, ${\sigma'}_{|T_A} = {\mu'}_{|T_A}$ and ${\sigma'}_{|T_B} = {\nu'}_{|T_B}$ $(N : m_0 + \overline{m}_0 \xrightarrow{\sigma'} m')$ and $m_k \leq m'$, and $\overline{m}_l \leq m'$.

By the structure of $\rightarrowtail_G$, two cases need to be considered: $k < |\mu|$ and $\overrightarrow{\mu_{[1..k+1]}}_{|R_{out}} \leq \overrightarrow{\nu_{[1..l]}}_{|R_{out}}$ or (2) $l < |\nu|$ and $\overrightarrow{\nu_{[1..l+1]}}_{|R_{in}} \leq \overrightarrow{\mu_{[1..k]}}_{|R_{in}}$.

First suppose $k < |\mu|$ and $\overrightarrow{\mu_{[1..k+1]}}_{|R_{out}} \leq \overrightarrow{\nu_{[1..l]}}_{|R_{out}}$. Let $t = \mu(k{+}1)$. If $t \in R$, then $t \in T_B$. Hence, firing transition t does not change the internal marking of A, i.e. $m_{k|P_A} = m_{k+1|P_A}$. Choose $\sigma = \sigma'$ and $m = m'$. Then clearly the statement holds.

Otherwise, $t \notin R$. There are two cases to consider: either (a) $\overset{\bullet}{_N}t \cap G = \emptyset$ or (b) $\overset{\bullet}{_N}t \cap G \neq \emptyset$. If (a) $\overset{\bullet}{_N}t \cap G = \emptyset$, then $\overset{\bullet}{_N}t \leq m_{k|P_A} \leq m'$. Let $\sigma = \sigma'; \langle t \rangle$ and $m \in I\!N^{P_N}$ such that $(N : m' \xrightarrow{t} m)$. Hence, σ and m have the desired property. Next, suppose (b) $\overset{\bullet}{_N}t \cap G \neq \emptyset$. Then, transition t needs input from some places in the interface G. Since $t \notin R$, we have $\overrightarrow{\mu_{[1..k]}}_{|R_{out}} = \overrightarrow{\mu_{[1..k+1]}}_{|R_{out}} \leq \overrightarrow{\nu_{[1..l]}}_{|R_{out}} = \overrightarrow{\sigma'}_{|R_{out}}$. Let $p \in \overset{\bullet}{_N}t \cap G$ be an interface place in the preset of transition t. Since transition t is enabled in (N, m_k), we have $m_k(p) > 0$. By the marking equation, $m_k(p) = m_0(p) + \sum_{u \in {}^\bullet p} \overrightarrow{\mu_{[1..k]}}(u) - \sum_{u \in p^\bullet} \overrightarrow{\mu_{[1..k]}}(u)$. As place p is an interface place, $m_k(p) \leq m'(p)$. Thus, transition t is enabled in (N, m'). Let $\sigma = \sigma'; \langle t \rangle$ and $m \in I\!N^{P_N}$ such that $(N : m' \xrightarrow{t} m)$.

Suppose (2) $l < |\nu|$ and $\overrightarrow{\nu_{[1..l+1]}}_{|R_{in}} \leq \overrightarrow{\mu_{[1..k]}}_{|R_{in}}$. Let $t = \nu(l+1)$. If $\overset{\bullet}{_N}t \cap G = \emptyset$, then $\overset{\bullet}{_N}t \leq m_l$. Hence, the statement holds for $\sigma = \sigma'; \langle t \rangle$ and $m \in I\!N^{P_N}$ such that $(N : m' \xrightarrow{t} m)$.

Otherwise, $\overset{\bullet}{_N}t \cap G \neq \emptyset$. Then $\lambda_G(t) =?$ and transition t needs input from A in order to be enabled in N. Hence, $t \in R_{in}$ and $\nu_{[1..l]}{}_{|R_{in}}(t) < \mu_{[1..k]}{}_{|R_{in}}(t)$. Let $p \in \overset{\bullet}{_N}t \cap G$ be an interface place in the preset of t. Consequently, $m_k(p) < m'(p)$, and transition t is enabled in (N, m'). Let $\sigma = \sigma'; \langle t \rangle$ and $m \in I\!N^{P_N}$ such that $(N : m' \xrightarrow{t} m)$. □

Lemma 20 shows that a firing sequence and a firing sequence it is elastic to may be interweaved into a new firing sequence that is elastic to both sequences. As in elastic communication the number of occurrences of each communicating transition should be equal, we may directly conclude that Property 10 holds for the elastic sequence relation ψ.

Corollary 21 (Harlem shuffle). *Let A and B be two OPNs that are composable with respect to port $G \in \mathcal{G}_A \cap \mathcal{G}_B$. Let $N_1 = \mathcal{C}_B(A)$ and $N_2 = \mathcal{S}(B)$. Let $\mu \in T_{N_1}^*$ and $m, m' \in I\!N^{P_{N_1}}$ such that $(N_1 : m \xrightarrow{\mu} m')$. Let $\nu \in T_{N_2}^*$ and $\overline{m}, \overline{m}' \in I\!N^{P_{N_2}}$ such that $(N_2 : \overline{m} \xrightarrow{\nu} \overline{m}')$ and $\psi_G(\mu, \nu)$. Then, there exists $\sigma \in T_N^*$ such that $(\mathcal{S}(N) : m + \overline{m} \xrightarrow{\sigma} m' + \overline{m}')$, $\psi_G(\mu, \sigma)$ and $\psi_G(\nu, \sigma)$.*

From Corollaries 18 and 21 we can directly conclude that Condition com$_\psi$ is a sufficient condition for compositional verification.

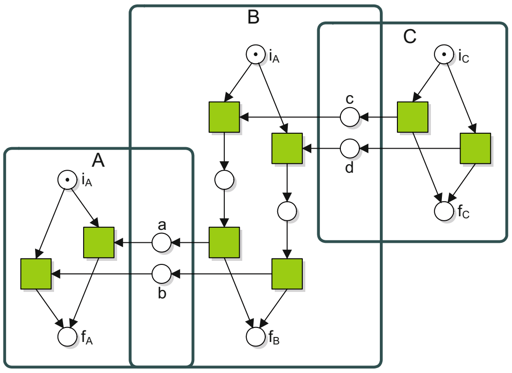

Fig. 7. Although net $A \oplus_G B \oplus_H C$ is sound, condition $\Psi_G(B, C)$ does not hold.

Theorem 22 (Elastic communication condition sufficient for soundness). *Let A, B and C be three OPNs such that A and B are composable with respect to* $G \in \mathcal{G}_A \cap \mathcal{G}_B$*, B and C are composable, A and C are disjoint and* $A \oplus_G B$ *is sound. If* $\text{com}_{\psi_G}(B, C)$ *holds, then* $A \oplus_G B \oplus_H C$ *is sound.*

The framework does not provide a necessary condition. As shown in Fig. 7, also the elastic communication condition is not necessary. In this example, component A either receives an a or a b from component B. In the composition $B \oplus_H C$, component C decides which message will be sent by component B. Consider the marking $[i_B, d, f_C]$ of the composition $B \oplus C$. In this marking, the composition can only decide to send message b, whereas if we project this marking on B, i.e. we consider only the marking $[i_B]$, also message a could be sent. Hence, the condition does not hold for the example.

6 Conclusions

In this paper, we considered a sub class of dynamic networks of asynchronously comunicating systems. We presented a framework for compositional verification of such systems based on communication conditions.

The elastic communication condition is an example of using this framework. Given two sequences, the elastic communication condition allows transitions that send messages to occur earlier in the firing sequence, as long as it is produced before the token needs to be consumed. A simple algorithm exists to decide whether a firing sequence is elastic to another firing sequence, and if so, the algorithm returns an interweaved firing sequence of the two.

Related Work. In [7] the authors give a constructive method preserving the inheritance of behavior. As shown in [2] this can be used to guarantee the correctness of interorganizational processes. Other formalisms, like I/O automata [14] or interface automata [6] use synchronous communication, whereas we focus on asynchronous communication.

In [23], the author introduces place composition to model asynchronous communication focusing on the question which subnets can be exchanged such that the behavior of the whole net is preserved. In [13] the authors focus on deciding *controllability* of an OPN and computing its *operating guidelines*. Operating guidelines can be used to decide substitutability of services [21], or to prove that an implementation of a service meets its specification [5].

In [8], the authors propose to model choreographies using Interaction Petri nets. Similarly the authors of [11] propose a method to verify whether services agree to a choreography specification. However, in these approaches the whole network should be known at design-time.

In [9], the authors introduce an abstract component and interface algebra based on logic, where consistency is based on the composition of, possibly infinite, sets of traces of both the connections and the services. Although closely related, the approach presented in this paper focuses more on the process aspects of component-based design.

Future Work. Although we have shown that the elastic communication condition is sufficient, decidability of the condition remains future work. The proposed framework shows that post-design verification is a challenging task. As, in limitations one first shows oneself the master, we search for similar approaches as presented in [12] that guarantees the presented conditions during the construction of a network of asynchronously communication systems.

References

1. van der Aalst, W.M.P.: Verification of workflow nets. In: Azéma, P., Balbo, G. (eds.) ICATPN 1997. LNCS, vol. 1248, pp. 407–426. Springer, Heidelberg (1997)
2. van der Aalst, W.M.P.: Inheritance of interorganizational workflows: how to agree to disagree without loosing control? Inf. Technol. Manage. J. **4**(4), 345–389 (2003)
3. van der Aalst, W.M.P., Beisiegel, M., van Hee, K.M., König, D., Stahl, C.: An SOA-based architecture framework. Int. J. Bus. process Integr. Manage. **2**(2), 91–101 (2007)
4. van der Aalst, W.M.P., van Hee, K.M., Massuthe, P., Sidorova, N., van der Werf, J.M.: Compositional service trees. In: Franceschinis, G., Wolf, K. (eds.) PETRI NETS 2009. LNCS, vol. 5606, pp. 283–302. Springer, Heidelberg (2009)
5. van der Aalst, W.M.P., Lohmann, N., Massuthe, P., Stahl, C., Wolf, K.: Multiparty contracts: agreeing and implementing interorganizational processes. Comput. J. **53**(1), 90–106 (2010)
6. de Alfaro, L., Henzinger, T.A.: Interface automata. SIGSOFT Softw. Eng. Notes **26**(5), 109–120 (2001)
7. Basten, T., van der Aalst, W.M.P.: Inheritance of behavior. J. Logic Algebraic Program. **47**(2), 47–145 (2001)
8. Decker, G., Weske, M.: Local enforceability in interaction petri nets. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 305–319. Springer, Heidelberg (2007)

9. Fiadeiro, J.L., Lopes, A.: An interface theory for service-oriented design. Theor. Comput. Sci. **503**, 1–30 (2013)
10. van Glabbeek, R.J.: The linear time - branching time spectrum II. In: Best, E. (ed.) CONCUR 1993. LNCS, vol. 715, pp. 66–81. Springer, Heidelberg (1993)
11. Gössler, G., Salaün, G.: Realizability of choreographies for services interacting asynchronously. In: Arbab, F., Ölveczky, P.C. (eds.) FACS 2011. LNCS, vol. 7253, pp. 151–167. Springer, Heidelberg (2012)
12. van Hee, K.M., Sidorova, N., van der Werf, J.M.: Construction of asynchronous communicating systems: weak termination guaranteed!. In: Baudry, B., Wohlstadter, E. (eds.) SC 2010. LNCS, vol. 6144, pp. 106–121. Springer, Heidelberg (2010)
13. Lohmann, N., Massuthe, P., Wolf, K.: Operating guidelines for finite-state services. In: Kleijn, J., Yakovlev, A. (eds.) ICATPN 2007. LNCS, vol. 4546, pp. 321–341. Springer, Heidelberg (2007)
14. Lynch, N.A., Tuttle, M.: Hierarchical correctness proofs for distributed algorithms. In: 6th Annual ACM Symposium on Principles of Distributed Computing (1987)
15. Massuthe, P.: Operating guidelines for services. Ph.D. thesis, Technische Universiteit Eindhoven (2009)
16. Massuthe, P., Serebrenik, A., Sidorova, N., Wolf, K.: Can I find a partner? Undecidability of partner existence for open nets. Inf. Process. Lett. **108**(6), 374–378 (2008)
17. McIlroy, M.D.: Mass produced software components. In: Naur, P., Randell, B. (eds.) Proceedings of NATA Software Engineering Conference, Garmisch Germany, vol. 1, pp. 138–150 (1968)
18. Papazoglou, M.P.: Web Services: Principles and Technology. Pearson/Prentice Hall, Tappan (2007)
19. Reisig, W.: Petri Nets: An Introduction. Monographs in Theoretical Computer Science: An EATCS Series, vol. 4. Springer, Berlin (1985)
20. Stahl, C.: Service substitution. Ph.D. thesis, Technische Universiteit Eindhoven (2009)
21. Stahl, C., Massuthe, P., Bretschneider, J.: Deciding substitutability of services with operating guidelines. In: Jensen, K., van der Aalst, W.M.P. (eds.) Transactions on Petri Nets and Other Models of Concurrency II. LNCS, vol. 5460, pp. 172–191. Springer, Heidelberg (2009)
22. Szyperski, C.: Component Software - Beyond Object-Oriented Programming. Addison-Wesley/ACM Press, New York (1998)
23. Vogler, W.: Asynchronous communication of petri nets and the refinement of transitions. In: Kuich, W. (ed.) Automata, Languages and Programming. LNCS, vol. 623, pp. 605–616. Springer, Heidelberg (1992)
24. van der Werf, J.M.E.M.: Compositional design and verification of component-based information systems. Ph.D. thesis, Technische Universiteit Eindhoven (2011)
25. Wolf, K.: Does my service have partners? In: Jensen, K., van der Aalst, W.M.P. (eds.) Transactions on Petri Nets and Other Models of Concurrency II. LNCS, vol. 5460, pp. 152–171. Springer, Heidelberg (2009)
26. Wolf, K., Stahl, C., Weinberg, D., Ott, J., Danitz, R.: Guaranteeing weak termination in service discovery. Fundam. Inf. **108**(1–2), 151–180 (2011)