Handshake Languages do not Compose

Luca Fossati[1] and Daniele Varacca[2]

[1]Queen Mary University of London [2] PPS - Université Paris Diderot & CNRS

Abstract. The handshake protocol is an asynchronous communication protocol that enforces several properties such as absence of computation interference and insensitivity from delays of propagation on wires. One of the simplest formalisations of the handshake protocol is via sets of traces (languages). However, such a model still lacks a satisfactory notion of composition. In this paper we propose a minimal set of axioms and show that there does not exist any notion of composition which respects them all. The relevance of this finding goes beyond the scope of the handshake protocol, since the impossibility result applies to any similar model of asynchronous systems which relaxes the restrictions enforced in handshake communications.

1 Introduction

The handshake protocol is a protocol of asynchronous communication between concurrent systems. It is designed to guarantee the absence of both *computation-interference* (i.e. that no message is sent to an unready receiver) and *transmission-interference* (i.e. that no channel carries more than one message) by requiring each message to be followed by a message of confirmation (*request* and *acknowledge*). It has several practical applications, for instance it has been employed in the implementation of VLSI chips (see, e.g. [9]).

The formal analysis of the handshake protocol was initiated by Van Berkel [17] who proposed a *trace model*. He considered the set of finite traces which describe the communications of a system abiding to the handshake protocol. We will call such sets *handshake languages*[1], and their elements *handshake traces*. Besides the alternance of requests and acknowledges on each channel, handshake languages enjoy other properties which model the fact that systems are receptive and behave correctly in asynchronous environments. Handshake languages are not prefix-closed, as the traces they contain are meant to represent the points of a communication where the system is *quiescent* i.e. where it waits for further inputs. This allows to represent different forms of nondeterminism and deadlocks.

Since systems are meant to interact with one another, any formal model must come equipped with a notion of *composition*. The notion of composition proposed by Van Berkel [17] looks natural and intuitive but, as the first author [4] has shown, it is not associative. To overcome this problem, other models were proposed. One solution was to to limit the amount of nondeterminism [3]. A Petri nets model [6] and a CCS-inspired process calculus [5] were subsequently designed (both complete with respect to the languages in [17]) along with a fully-abstract interpretation from the calculus' terms to nets.

[1] Van Berkel used the term "handshake circuit" since his aim was the application to VLSI design.

However one question remains open. Is it possible to define a trace-based semantics of the handshake protocol which:

1. is sound and complete with respect to handshake behaviours
2. can represent nondeterminism and deadlocks, and
3. contains an associative notion of composition which respects some basic constraints?

The requirements (1) and (2) imply that we need to consider languages which adhere to the conditions requested in the original model ([17]). Then, we need to state which are the "basic constraints" that a good notion of composition should satisfy (3).

In this paper we propose *two* constraints that any good notion of composition should satisfy. Intuitively, we want the composed language to contain precisely the traces that can be observed when the two languages involved in the composition interact. Therefore we *require* it to contain all the traces that result from an external observation of an interaction in which both systems execute a quiescent trace.

However we also need to take into account the *infinite internal chatterings* taking place, i.e. those situations in which the two languages are never quiescent at the same time and yet engage in an infinite internal communication that never produces any observable message. Therefore, we also *allow* the composed language to possibly include any trace which is *potentially* the external observation of some infinite internal chattering situation.

To resume, a composition is:

– required to contain every trace that is definitely observable;
– allowed to contain some additional traces that are potentially observable.

The main result of the paper is that there is no notion of composition that satisfies these two requirements and that at the same is associative. Since the requirements are independent from specific features of the handshake protocol, our negative result applies to any larger class of languages as well. In § 5 we further discuss and provide more detailed context to these implications.

The paper is organised as follows. In § 2, we introduce handshake languages and their theoretical framework. In § 3, we provide a minimal set of axioms that any notion of composition should satisfy. In § 4, we derive additional properties from the axioms and show that there does not exist any notion of composition that satisfies them all. Finally, we conclude in § 5.

2 Alphabets, Traces and Languages

In this section we recall the definition of handshake alphabets and handshake languages. Handshake alphabets add some structure over the symbols set, laying the basis for defining the properties of handshake languages.

2.1 Channels, Ports, Alphabets and Traces

According to the handshake paradigm, systems communicate over bidirectional channels in alternation. Channels are denoted by $a, b, c, \ldots$, while $a^?$ and $a^!$ denote, respectively, the input and the output message that a system can receive (respectively, send) over channel a. We write $Msg(a)$ to indicate the set $\{a^?, a^!\}$. The uniqueness of such messages corresponds to the assumption that no data is exchanged in a communication: we only

consider the bare synchronisation signals, regardless of what information they might contain. This does simplify the traditional notion of handshake communication but it does not make our result less general. In fact, a negative result which holds for a sub-class of handshake languages, automatically holds for the larger class as well.

A system's interface towards a channel is called port and is defined as follows.

Definition 2.1 (handshake port). A *handshake port* $A \stackrel{\text{def}}{=} \langle a, \lambda \rangle$ consists of a channel a and a polarity $\lambda \in \{+, -\}$, which may be *positive* $(+)$ or *negative* $(-)$.

Handshake ports are the simplest forms of *handshake alphabets*. Given a port $A = \langle a, \lambda \rangle$, we can only observe communication traces of the following forms:

$$a^! a^? a^! a^? \ldots \qquad (\text{if } \lambda = +)$$
$$a^? a^! a^? a^! \ldots \qquad (\text{if } \lambda = -)$$

a *sequential handshake trace* (on A) is any finite trace of such shape. We call *requests* the message occurrences at odd positions in a sequential handshake trace (e.g. $a^!$ if $\lambda = +$), and *acknowledges* those at even positions (e.g. $a^?$ if $\lambda = +$). We call *handshake* any pair of occurrences consisting of a request and the successive acknowledge in a sequential handshake trace, e.g. those underlined (and also those not underlined) in the traces below:

$$a^! a^? \underline{a^! a^?} a^! a^? \underline{a^! a^?} a^! a^? \underline{a^! a^?} a^! a^? \qquad \underline{a^? a^!} a^? a^! \underline{a^? a^!} a^? a^! \underline{a^? a^!} a^? a^! \underline{a^? a^!}$$

The port A admits a *dual port* $A^\perp = \langle a, \lambda^* \rangle$, where $+^* = -$ and $-^* = +$. Thus, dual ports share the same communication channel a but have dual polarities. They represent the two ports at opposite ends of a. Note that from any non-empty sequential handshake trace we can infer both the channel and the polarity of the port over which the trace is defined.

Alphabets are sets of ports. Below, $ch(A)$ denotes the channel of a handshake port A.

Definition 2.2 (handshake alphabet). A *handshake alphabet* $\mathcal{A} \stackrel{\text{def}}{=} \{A_1, \ldots, A_n\}$ is a set of ports $A_1, \ldots, A_n$ such that $ch(A_i) \neq ch(A_j)$, for all $i \neq j$.

We write A for the singleton alphabet $\{A\}$. The functions $ch(-)$ and $Msg(-)$ are extended to general alphabets as follows:

$$ch(\mathcal{A}) = \{ch(A) \mid A \in \mathcal{A}\} \qquad Msg(\mathcal{A}) = \cup_{a \in ch(\mathcal{A})} Msg(a)$$

Given a handshake alphabet $\mathcal{A}$, let s be a sequence of symbols from the set $Msg(\mathcal{A})$ and let $\mathcal{A}' \subseteq \mathcal{A}$. The *projection of s on $\mathcal{A}'$*, denoted $s \restriction \mathcal{A}'$, is obtained by removing from s the symbols which are not in $Msg(\mathcal{A}')$. We define handshake traces as follows:

Definition 2.3 (handshake trace). A *handshake trace* on a handshake alphabet $\mathcal{A}$ is a sequence s of symbols from the set $Msg(\mathcal{A})$ such that, for any $A \in \mathcal{A}$, $s \restriction A$ is a sequential handshake trace. We write $\mathcal{L}_\mathcal{A}$ for the set of finite handshake traces over $\mathcal{A}$.

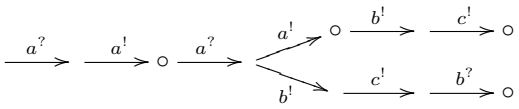
Henceforth we shall speak of "traces", always meaning handshake traces.

Note that, while the traces observable on handshake ports are constrained to one of two possible forms, handshake alphabets in general allow the observation of a variety of traces, since communications over different ports occur in parallel. A single handshake on a channel a may be interleaved with sequences of requests and acknowledges on different channels. A sequential handshake trace is thus a special case of a handshake trace.

2.2 Handshake Languages

We define a *language* as a set of finite handshake traces on a handshake alphabet $\mathcal{A}$, i.e. a subset of $\mathcal{L}_{\mathcal{A}}$. We let $\sigma, \tau, \rho, \ldots$ range over languages. We have already mentioned that from a non-empty sequential trace we can infer the port over which it is defined. A straightforward consequence is that we can infer the minimal alphabet on which a language is defined[2]. Then we write $\mathcal{A}_\sigma$ to denote such alphabet.

Languages can be conveniently seen as forest-like graphs [17]. For example, the language $\sigma_{bad} = \{a^?a^!, a^?a^!a^?a^!, a^?a^!a^?a^!b^!c^!, a^?a^!a^?b^!c^!b^?\}$ corresponds to the graph:



Since languages are in general not prefix-closed, our graphs contain small circles to indicate the prefixes which are actual traces of the language. Such prefixes represent the points of an execution in which the language may decide to stop (e.g. to wait for further inputs). For example, σ_{bad} may decide to stop at $a^?a^!a^?a^!$ and wait indefinitely *or* to go on and output a message on b. Thus, for any σ, $s \in \sigma$ is said to be a *quiescent trace* of σ.

Not all languages are handshake languages. Next we set the conditions for languages to be handshake languages. Let us start by introducing a few notational conventions. We write $s \sqsubseteq t$ to indicate that the trace s is a prefix of the trace t, we write $s \cdot t$ to denote the concatenation of s and t, and we write $\sigma^{\leq}$ to denote the prefix-closure of a language σ.

Given $s \in \sigma^{\leq}$, a prefix of some trace of σ, we say that s is *passive* (w.r.t. σ) if it cannot be extended with outputs in σ. Formally, s is passive iff:

$$\forall t \in \sigma \text{ s.t. } s \sqsubseteq t, \exists a \in ch(\mathcal{A}_\sigma) \text{ s.t. } t = s \cdot a^? \cdot t'$$

$\sigma^{\mathbf{P}}$ denotes the passive prefix-closure of σ (the union of σ and its set of passive prefixes).

Note that, at a passive trace, a language has no choice but to stop and wait for further inputs. Then we want all passive traces to be quiescent. Formally, as a first condition, we demand that $\sigma = \sigma^{\mathbf{P}}$ (σ is *closed under passive prefixes*).

The language σ_{bad} defined above is not closed under passive prefixes. For example the empty trace ϵ is passive w.r.t. σ_{bad} because it can only be extended with an input on a, and yet $\epsilon \notin \sigma_{bad}$. By closing σ_{bad} under passive prefixes we obtain:

$$(\sigma_{bad})^{\mathbf{P}} = \{\epsilon, a^?a^!, a^?a^!a^?a^!, a^?a^!a^?a^!b^!c^!, a^?a^!a^?b^!c^!, a^?a^!a^?b^!c^!b^?\}$$

Next we define a relation $\mathbf{r}_{\mathcal{A}}$ between traces on alphabet $\mathcal{A}$. We first define $\mathbf{r}_{\mathcal{A}}$ on traces consisting of only two *independent messages*, where messages are independent if they belong to two distinct channels. Let $a, b \in ch(\mathcal{A})$, s.t. $a \neq b$. Then we set:

$$a^!b^! \,\mathbf{r}_{\mathcal{A}}\, b^!a^! \qquad\qquad a^?b^? \,\mathbf{r}_{\mathcal{A}}\, b^?a^? \qquad\qquad a^?b^! \,\mathbf{r}_{\mathcal{A}}\, b^!a^?$$

Further, we close $\mathbf{r}_{\mathcal{A}}$ by concatenation, reflexivity and transitivity. If no confusion may arise, we write just $s \,\mathbf{r}\, t$, which reads s *reorders* t. Intuitively, whenever we find two independent outputs one after the other, we can reorder them by permuting their occurrences,

[2] The inferred alphabet is minimal because we can always add further unused ports to it.

and similarly for two independent inputs. But also, when an input follows an independent output, the trace can be reordered by moving the input back before the output.

A language σ is *reorder-closed* iff $s \in \sigma$ and $t\,\mathbf{r}\,s$ imply $t \in \sigma$. We write $\sigma^{\mathbf{r}}$ for the reorder-closure of σ. As a second condition, we ask that σ be reorder-closed: $\sigma = \sigma^{\mathbf{r}}$.

This requirement is common to most models of asynchronous communication. But, once again, our language σ_{bad} does not satisfy it. The following is its reorder-closure:

$$(\sigma_{bad})^{\mathbf{r}} = \{\, a^?a^!, a^?a^!a^?a^!, a^?a^!a^?a^!b^!c^!, a^?a^!a^?b^!a^!c^!, a^?a^!a^?b^!c^!a^!, a^?a^!a^?c^!b^!a^!,$$
$$a^?a^!a^?c^!a^!b^!, a^?a^!a^?b^!c^!b^?, a^?a^!a^?c^!b^!b^?, a^?a^!a^?b^!b^?c^! \,\}$$

We now define another relation $\mathbf{x}_{\mathcal{A}}$ as the smallest preorder on traces such that: for any $s \in \mathcal{L}_{\mathcal{A}}$ and $a \in ch(\mathcal{A})$, if $s \cdot a^? \in \mathcal{L}_{\mathcal{A}}$ then $s \cdot a^?\,\mathbf{x}_{\mathcal{A}}\,s$. Again, if no confusion may arise, we write $s\,\mathbf{x}\,t$, which reads: s *input-extends* t. The *input-extension closure* of a language σ, denoted $\sigma^{\mathbf{x}}$, is defined to contain all the traces of σ, plus those traces which input-extend some prefix of σ but which are not prefixes of σ themselves. As follows:

$$\sigma^{\mathbf{x}} = \sigma \cup \{t \mid t \notin \sigma^{\leq} \wedge \exists s \in \sigma^{\leq} \text{ s.t. } t\,\mathbf{x}_{\mathcal{A}_\sigma}\,s\}$$

The last condition we impose is that $\sigma = \sigma^{\mathbf{x}}$. When a language σ satisfies this condition we say that it is *receptive*. Intuitively, a language is receptive ports are always ready to accept an input, provided the alternation of input and output is respected. We leave it to the reader to figure out what the language $\sigma_{good} = (\sigma_{bad})^{\mathbf{pxr}} = (((\sigma_{bad})^{\mathbf{p}})^{\mathbf{x}})^{\mathbf{r}}$ looks like.

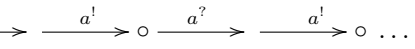
To resume, handshake language are defined as follows.

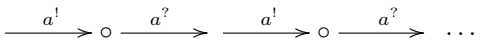
Definition 2.4 (handshake language). Let σ be a non-empty language. Then σ is a *handshake language* if it satisfies the following conditions:

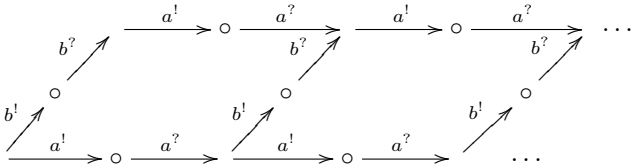
1. $\sigma = \sigma^{\mathbf{p}}$ (σ is *closed under passive prefixes*);
2. $\sigma = \sigma^{\mathbf{r}}$ (σ is *reorder-closed*);
3. $\sigma = \sigma^{\mathbf{x}}$ (σ is *receptive*).

We write HsLng to denote the set of handshake languages.

Henceforth we may speak of languages always referring to handshake languages. Here are three languages that shall be useful in the remaining of the paper.

$SKIP_a =$ 

$RUN_a =$ 

$HAC_{ab} =$ 

RUN_a is initially willing to send a request, then it becomes quiescent until it receives an acknowledge and starts again. $SKIP_a$ instead is initially waiting for a request and,

whenever one comes, it acknowledges it and goes back to wait. These two languages are inspired by [17], although Van Berkel's definitions were more general. HAC_{ab} is new (it stands for "handshake and continue"). Like RUN_a it can keep sending requests on a. Unlike RUN_a, it may decide to send a request on b instead (in a mutual exclusive way). After the handshake on b completes, it behaves deterministically like RUN_a.

The above languages, as well as others we shall introduce later on, are merely some of the simplest languages we could find that, combined in certain ways, allow us to expose the problems of composition. We did not choose them because of their computational relevance.

3 Handshake Composition

In this section we recall the notion of composable alphabets and propose a minimal set of axioms for the composition of two languages on composable alphabets.

3.1 Composable Alphabets and Interaction Traces

In order for two languages to be composable, they should be able to communicate on all common channels, while channels that are not used for communication should be used by either language but not by both. This intuition is formalised by the notion of composable alphabets. Consider an alphabet $\mathcal{A}$ and a channel $a \in ch(\mathcal{A})$. By definition, $\mathcal{A}$ contains exactly one port with channel a. Then we write $A_{a,\mathcal{A}}$ to denote the only port such that $A_{a,\mathcal{A}} \in \mathcal{A}$ and $a \in ch(A_{a,\mathcal{A}})$. Two alphabets $\mathcal{A}_1$ and $\mathcal{A}_2$ are *composable* if and only if $A_{a,\mathcal{A}_1} = (A_{a,\mathcal{A}_2})^{\perp}$, for all $a \in ch(\mathcal{A}_1) \cap ch(\mathcal{A}_2)$.

The duality between the two ports at the opposite ends of a communication channel is reflected in the traces on these ports: outputs on one side become inputs on the other, and viceversa. This may be a source of confusion if we want to interleave two traces on composable alphabets, which is often the case when dealing with language composition. One way to avoid the confusion is by renaming the internal messages, as we show next.

Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be composable alphabets. For all $a \in ch(\mathcal{A}_1) \cup ch(\mathcal{A}_2)$, we define $Msg_{\mathcal{A}_1,\mathcal{A}_2}(a)$ as a variant of $Msg(a)$ such that: if $a \in ch(\mathcal{A}_1) \cap ch(\mathcal{A}_2)$, then $Msg_{\mathcal{A}_1,\mathcal{A}_2}(a) = \{a^{\mathcal{A}_1}, a^{\mathcal{A}_2}\}$; otherwise, $Msg_{\mathcal{A}_1,\mathcal{A}_2}(a) = Msg(a) = \{a^?, a^!\}$. Basically, the variant consists in renaming internal messages, where $a^{\mathcal{A}}$ represents an output from $\mathcal{A}$'s side. We extend the function $Msg_{\mathcal{A}_1,\mathcal{A}_2}(-)$ to subsets of $\mathcal{A}_1 \cup \mathcal{A}_2$ as done for $Msg()$: for all $\mathcal{A}' \subseteq \mathcal{A}_1 \cup \mathcal{A}_2$, $Msg_{\mathcal{A}_1,\mathcal{A}_2}(\mathcal{A}') = \cup_{a \in ch(\mathcal{A})} Msg_{\mathcal{A}_1,\mathcal{A}_2}(a)$. We write $Msg_{\mathcal{A}_1,\mathcal{A}_2}$ to indicate $Msg_{\mathcal{A}_1,\mathcal{A}_2}(\mathcal{A}_1 \cup \mathcal{A}_2)$.

Now, consider a sequence s of symbols from $Msg_{\mathcal{A}_1,\mathcal{A}_2}$. We define $s \upharpoonright \mathcal{A}_1$, the *projection of* s on $\mathcal{A}_1$, by modifying s as follows: 1) remove all $ch(\mathcal{A}_2) \setminus ch(\mathcal{A}_1)$ symbols from s; 2) for all $a \in ch(\mathcal{A}_1) \cap ch(\mathcal{A}_2)$, rename all occurrences of $a^{\mathcal{A}_1}$ by $a^!$ and all occurrences of $a^{\mathcal{A}_2}$ by $a^?$. Similarly, we define the projection of s on subsets of $\mathcal{A}_1$ or of $\mathcal{A}_2$ (e.g. on a single port). Interleavings are defined similarly to traces (Definition 2.3):

Definition 3.1 (interleaving). An *interleaving* s on composable alphabets $\mathcal{A}_1$ and $\mathcal{A}_2$ is a sequence of symbols from $Msg_{\mathcal{A}_1,\mathcal{A}_2}$ such that:

– for any $a \in ch(\mathcal{A}_1)$ and $b \in ch(\mathcal{A}_2)$, $s \upharpoonright A_{a,\mathcal{A}_1}$ and $s \upharpoonright A_{b,\mathcal{A}_2}$ are sequential traces.

We write $\mathcal{L}_{\mathcal{A}_1,\mathcal{A}_2}$ to denote the set of all interleavings on $\mathcal{A}_1$ and $\mathcal{A}_2$.

3.2 Axioms for Composition

Interleavings represent all the traces that could ever possibly arise from the interaction of any pair of languages on their respective composable alphabets. It follows that any notion of composition between two languages σ and τ on composable alphabets should consider a subset of the interleavings on $\mathcal{A}_\sigma$ and $\mathcal{A}_\tau$.

The question is: which subset of interleavings should we retain? Intuitively, among all the interleavings on the two alphabets, only those that somehow arise from the interaction between σ and τ should be relevant. Indeed a minimal condition could be that the projection of an interleaving on $\mathcal{A}_\sigma$ be in $\sigma^{\leq}$ and that its projection on $\mathcal{A}_\tau$ be in $\tau^{\leq}$. But that is not enough, quiescence should be taken into account as well.

How can we decide if a trace, obtained through external projection of an interleaving, is quiescent or not? We would expect that σ and τ "agree" upon some interleaving, which happens when they simultaneously reach a quiescent trace. But this excludes one scenario, which occurs when the two languages engage in an *infinite internal chat*, where they keep exchanging internal messages and forever avoid to output on the outside of the composition. Then it could well be that, in this infinite internal chat, they never reach a quiescent trace simultaneously. But since no external output occurs either, we should consider the external projections of these infinite interleavings as quiescent as well. Let us formalise the above intuitions.

First, we define the *weave* of two languages as the set of all the interleavings in which both languages have simultaneously reached a quiescent trace. Formally:

Definition 3.2 (weave). We define the *weave* of σ and τ (denoted by $\sigma \,\mathbf{w}\, \tau$) as follows:

$$\sigma \,\mathbf{w}\, \tau = \{s \in \mathcal{L}_{\mathcal{A}_1,\mathcal{A}_2} \mid s \upharpoonright \mathcal{A}_\sigma \in \sigma \wedge s \upharpoonright \mathcal{A}_\tau \in \tau\}$$

Interleavings that may be extended with an infinite internal chat are called *divergences* and are defined below. In the following definition, we use the notation S^ω to denote the set of all the infinite sequences of symbols from S (for any set of symbols S). We also write $u' \sqsubset u$ to indicate that u' is a strict prefix of u (it follows that u' must be finite, independently of u being finite or infinite).

Definition 3.3 (divergences set). We define *the set of divergences of σ and τ* (denoted by $\sigma \,\mathbf{div}\, \tau$) as follows:

$$\sigma \,\mathbf{div}\, \tau = \left\{ s \in \mathcal{L}_{\mathcal{A}_\sigma,\mathcal{A}_\tau} \mid \exists u \in (Msg_{\mathcal{A}_\sigma,\mathcal{A}_\tau}(\mathcal{A}_\sigma \cap \mathcal{A}_\tau))^\omega, \forall u' \sqsubset u, s \cdot u' \in \sigma^{\leq} \,\mathbf{w}\, \tau^{\leq} \right\}$$

The above definition of divergence was chosen on purpose to be as weak as possible. In fact, we do not expect that all these divergences can *really* be extended with infinite internal chats: for instance, some internal chats may be indefinitely long, but always finite. Indeed, the search for the "good composition" can be seen as an attempt to distinguish between internal chats that can grow indefinitely long but are bound to end, and those that can actually go on forever. In chats of the former kind, one of the two languages at some point deviates from the infinite internal chat and perhaps outputs an external message. As we shall show in the rest of this paper, there is no criterion to tell apart these chats from real infinite internal ones.

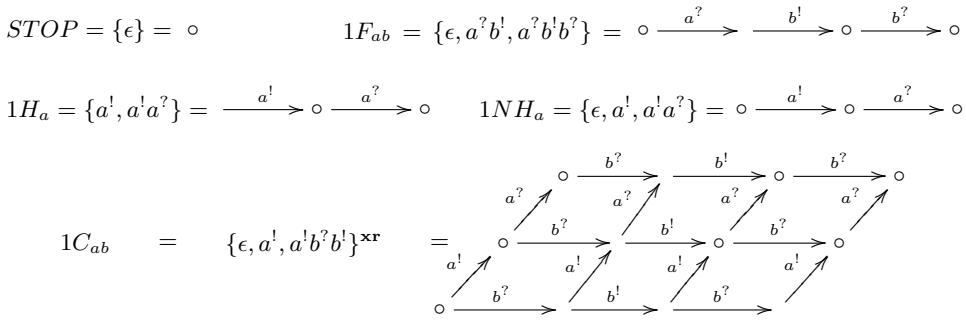


Fig. 1. 1-handshake languages used in the proof

Now, let $\mathcal{A}^{Ext}_{\sigma,\tau} = \{\mathcal{A}_{a,\sigma} \mid a \in (ch(\mathcal{A}_\sigma) \backslash ch(\mathcal{A}_\tau))\} \cup \{\mathcal{A}_{a,\tau} \mid a \in (ch(\mathcal{A}_\tau) \backslash ch(\mathcal{A}_\sigma))\}$ denote the external restriction of the alphabets of two languages σ and τ. The following definition formalises the criteria for language composition introduced above.

Definition 3.4 (axioms). Let $- \circ - : \mathsf{HsLng} \times \mathsf{HsLng} \to \mathsf{HsLng}$ be a function taking a pair of handshake languages and returning another handshake language. In order to be a *composition between handshake languages*, $- \circ -$ must satisfy the following axioms, for all handshake languages σ, τ and ρ:

1. $(\sigma \circ \tau) \circ \rho = \sigma \circ (\tau \circ \rho)$ (associativity);
2. $\sigma \circ \tau \subseteq (\sigma \,\mathbf{w}\, \tau \cup \sigma \,\mathbf{div}\, \tau) \restriction \mathcal{A}^{Ext}_{\sigma,\tau}$ (soundness);
3. $(\sigma \,\mathbf{w}\, \tau) \restriction \mathcal{A}^{Ext}_{\sigma,\tau} \subseteq \sigma \circ \tau$ (completeness).

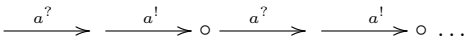
Associativity is a basic requirement. Soundness and completeness together require that *all* interleavings in the weave and, possibly, *some* divergences should contribute to define composition. Implicitly they also require that nothing else should.

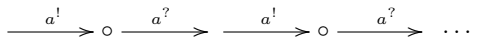
4 The Impossibility Result

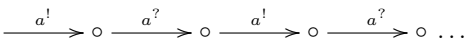
We start by assuming that some composition $- \circ - : \mathsf{HsLng} \times \mathsf{HsLng} \to \mathsf{HsLng}$ satisfies all the axioms given in Definition 3.4. Then, through a series of successive results, we reach a condtraticton implying that $\circ$ cannot exist. The intermediate results are meant to constrain $\circ$ when given specific pairs of languages as arguments. We start in § 4.1 by listing all the languages that we use in the proof and establishing a series of immediate results. Then in § 4.2, we do the proof. The two subsections can be either read in the order, or starting from § 4.2 and going back to § 4.1 each time a new language is encountered.

4.1 Introducing the Languages

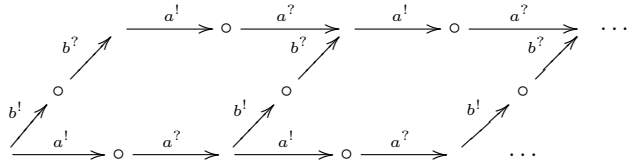
Figure 1 defines all the finite languages that shall be used in the proof. The advantage of finite languages is that they never engage in infinite chatterings. Thus we shall use them in decomposing complex languages into simpler ones. The languages in Figure 1 are not only finite but also very small sets. We call them 1-*handshake languages* (or simply 1-*languages*), because they engage at most in one handshake per channel.

$SKIP_a = \{\epsilon, a^?a^!, a^?a^!a^?a^!, \ldots\} =$ 

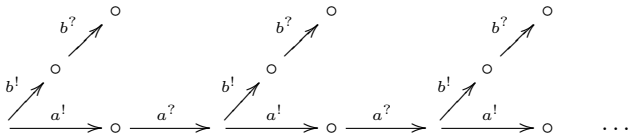
$RUN_a = \{a^!, a^!a^?a^!, \ldots\} =$ 

$CHAOS_a = \{\epsilon, a^!, a^!a^?, a^!a^?a^!, \ldots\} =$ 

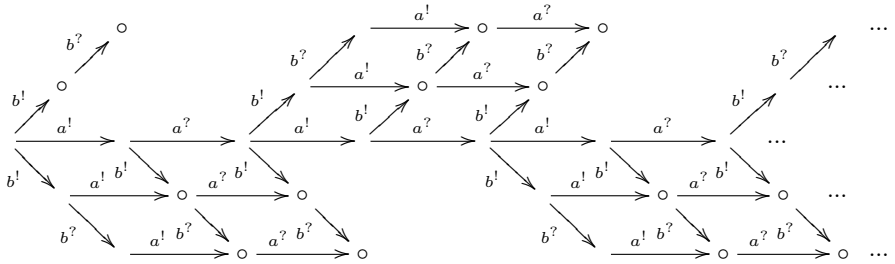
$HAC_{ab} = \{b^!, b^!b^?a^!, b^!b^?a^!a^?a^!, \ldots, a^!, a^!a^?b^!, \ldots, a^!a^?a^!, \ldots\} =$

$HAC_{ab} =$ 

$CES_{ab} = \{b^!, a^!, a^!a^?b^!, a^!a^?a^!, \ldots\}^{\mathbf{x}} =$



$PES_{ab} = \{b^!, b^!a^!, b^!a^!a^?, a^!a^?b^!a^!, a^!a^?b^!a^!a^?, \ldots\}^{\mathbf{xr}} =$



$EC_{abc} = \{\{c^!, a^!, a^!a^?c^!b^!, a^!a^?c^!b^!b^?, a^!a^?b^!b^?a^!, a^!a^?b^!b^?a^!a^?c^!b^!, \ldots\}^{\mathbf{xr}} =$

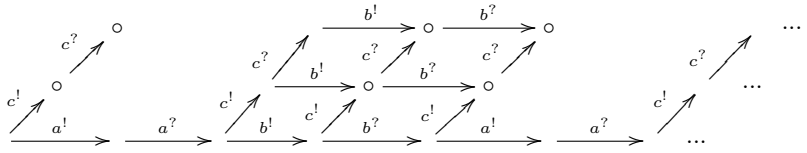


Fig. 2. Infinite handshake languages used in the proof

$STOP$ is the language which does nothing[3]. $1H_a$ outputs on channel a once and then waits for an acknowledgement (it does "one handshake"). $1NH_a$ ("one non-deterministic

[3] Adhering to our assumption of alphabet minimality, $STOP$ is defined on the empty alphabet.

handshake"). non-deterministically chooses whether to do the handshake or to wait (forever). The difference between $1H_a$ and $1NH_a$ is subtle ($\epsilon \notin 1H_a$ and $\epsilon \in 1NH_a$) but the impossibility proof rests largely on this subtle difference. $1F_{ab}$ ("one forwarding") forwards on b any request it receives from a, but it never replies on a. $1C_{ab}$ stands for "one-time confusion". Like $1NH_a$, it can also choose to send on a or to wait. But this choice can be resolved by the arrival of an input on a different channel b: once a request on b is received then the request on a must be sent (and also the acknowledge on b).

Figure 2 defines all the infinite languages that we shall use in the proof. For convenience, we have also included those that we introduced at the end of § 2 (RUN_a, $SKIP_a$ and HAC_{ab}). $CHAOS_a = RUN_a^{\leq}$ is the prefix closure of RUN_a. It can become quiescent at any time. CES_{ab} stands for "chattering with end signal". Every time it may send a request on channel a, it has a choice to either do it or end the communication on a, by sending a "signal" on b. PES_{ab} ("postponable end signal") can be best understood by comparison with CES_{ab}. In PES_{ab}, an output on channel b may either signal that the chattering on a has ended (like in CES_{ab}) or that it will end after one more handshake. Note that the choice between two distinct "$b^!$" branches (which can be seen in the graphical representation of Figure 2) is internal, and thus *not observable* in the language definition. EC_{abc} stands for "early choice". Periodically, it has a choice between sending an output on a or on c, but the choice may be done before the output on a is even enabled.

Before we move on to the impossibility proof, we state the following facts about the composition $\circ$, which follow immediately by application of Definition 3.2 (weave). In each of them, channels a, b and c are to be understood as arbitrary and distinct.

Fact 1 $\forall \sigma,\ STOP \circ \sigma = \sigma \circ STOP = \sigma$

Fact 2 $HAC_{ac} \circ 1C_{bc} = RUN_a \circ 1NH_b$

Fact 3 $CES_{ab} = HAC_{ac} \circ 1F_{cb}$

Fact 4 $SKIP_a \circ CHAOS_a = STOP$

Fact 5 $PES_{ab} \circ 1C_{cb} = CHAOS_a \circ 1H_c$

Fact 6 $SKIP_a \circ EC_{abc} = PES_{bc}$

Fact 7 $SKIP_b \circ EC_{abc} = CES_{ac}$

Fact 8 $CES_{ab} \circ 1F_{bc} = CES_{ac}$

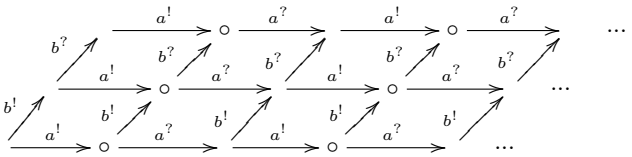
As illustration, Figure 3 shows the graph of the composition for Fact 2 and Fact 5.

Even though we have not required the symmetry of $\circ$ as an axiom, each composition in any of the facts above is symmetric, as the following lemma shows.

Lemma 4.1. *Let σ and τ be handshake languages s.t. $\sigma \,\mathbf{div}\, \tau = \emptyset$, then $\sigma \circ \tau = \tau \circ \sigma$.*

Proof. It follows from the symmetry of the weave (Definition 3.2). □

(Fact 2) $HAC_{ac} \circ 1C_{bc} = RUN_a \circ 1NH_b =$



(Fact 5) $PES_{ab} \circ 1C_{cb} = CHAOS_a \circ 1H_c =$

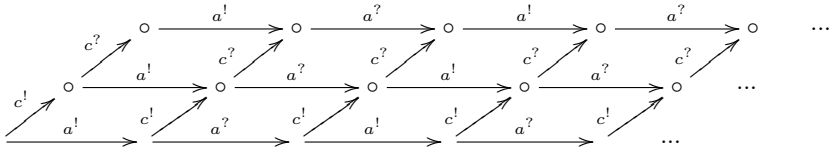


Fig. 3. Some compositions with empty set of divergences

4.2 Proving the Result

The proof of the impossibility goes as follows: we consider three specific languages σ, τ, ρ. Using associativity, soundness and completeness, we determine the languages $(\sigma \circ \tau) \circ \rho$ and $\sigma \circ (\tau \circ \rho)$ We show then that these two languages are different, which contradicts associativity.

The following is a basic lemma that is used in the proof.

Lemma 4.2. $RUN_a \circ SKIP_a = SKIP_a \circ RUN_a = STOP$

Proof. By definition of weave (Definition 3.2), we have $RUN_a \,\mathbf{w}\, SKIP_a = \emptyset$. Similarly, $(RUN_a \,\mathbf{div}\, SKIP_a) \upharpoonright \mathcal{A}^{Ext}_{RUN_a, SKIP_a} = \{\epsilon\}$ (Definition 3.3). Then soundness (axiom 2 of Definition 3.4) implies that $RUN_a \circ SKIP_a \subseteq \{\epsilon\}$. But handshake languages are non-empty (Definition 2.4). Then $RUN_a \circ SKIP_a = STOP = \{\epsilon\}$. Since $\mathbf{w}$ and $\mathbf{div}$ are symmetric operators, we show $SKIP_a \circ RUN_a = STOP$ in the same way. □

Lemma 4.2 relies on the non-emptiness of languages. In § A, we give a formal account of the inconsistency of the alternative (allowing languages to be empty). For now, observe that a language with no traces at all does not make sense if we want it to represent the *quiescent* executions (as opposed to a language containing only the empty trace).

Then we show:

Lemma 4.3. $SKIP_a \circ HAC_{ac} = HAC_{ac} \circ SKIP_a = 1NH_c$

Proof. Note that $c^! \in (SKIP_a \,\mathbf{w}\, HAC_{ac}) \upharpoonright \mathcal{A}^{Ext}_{SKIP_a, HAC_{ac}}$. It follows by completeness (axiom 3 of Definition 3.4) that $c^! \in SKIP_a \circ HAC_{ac}$. Then by receptiveness of handshake languages (condition 3 of Definition 2.4), we also have $c^! c^? \in SKIP_a \circ HAC_{ac}$. The question now is whether $\epsilon \in SKIP_a \circ HAC_{ac}$. To answer it, note that:

$$\begin{aligned} SKIP_a \circ (HAC_{ac} \circ 1C_{bc}) &= SKIP_a \circ (RUN_a \circ 1NH_b) && \text{(Fact 2)} \\ &= (SKIP_a \circ RUN_a) \circ 1NH_b && \text{(assoc. - Def. 3.4(1))} \\ &= STOP \circ 1NH_b = 1NH_b && \text{(Lemma 4.2, Fact 1)} \end{aligned}$$

First suppose that $\epsilon \notin SKIP_a \circ HAC_{ac}$, which implies $SKIP_a \circ HAC_{ac} = 1H_c$. Then $(SKIP_a \circ HAC_{ac}) \circ 1C_{bc} = 1H_c \circ 1C_{bc} = 1H_b$, where the last equivalence is because $1C_{bc}$ always outputs on b once it has received an input on c. However, this breaks associativity of composition (axiom 1 of Definition 3.4):

$$(SKIP_a \circ HAC_{ac}) \circ 1C_{bc} = 1H_b \neq 1NH_b = SKIP_a \circ (HAC_{ac} \circ 1C_{bc})$$

Then suppose $\epsilon \in SKIP_a \circ HAC_{ac}$, which implies $SKIP_a \circ HAC_{ac} = 1NH_c$. Now we have $(SKIP_a \circ HAC_{ac}) \circ 1C_{bc} = 1NH_c \circ 1C_{bc} = 1NH_b$, where the last equivalence is because $1NH_c$ does not guarantee that it will send a request on c. Then:

$$(SKIP_a \circ HAC_{ac}) \circ 1C_{bc} = 1NH_b = SKIP_a \circ (HAC_{ac} \circ 1C_{bc})$$

Thus we have $SKIP_a \circ HAC_{ac} = 1NH_c$. By applying a symmetric reasoning and Lemma 4.1, we also have $HAC_{ac} \circ SKIP_a = 1NH_c$. □

The following lemma is the main result for one direction of the impossibility proof.

Lemma 4.4. $SKIP_a \circ CES_{ab} = CES_{ab} \circ SKIP_a = 1NH_b$

Proof.

$$\begin{aligned} SKIP_a \circ CES_{ab} &= SKIP_a \circ (HAC_{ac} \circ 1F_{cb}) && \text{(Fact 3)} \\ &= (SKIP_a \circ HAC_{ac}) \circ 1F_{cb} && \text{(assoc. - Def. 3.4(1))} \\ &= 1NH_c \circ 1F_{cb} = 1NH_b && \text{(Lemma 4.3)} \end{aligned}$$

By a symmetric reasoning (and using Lemma 4.1 in the first and last equalities) we can also state $CES_{ab} \circ SKIP_a = 1NH_b$. □

The proof of the following lemma has the same structure as that of Lemma 4.3.

Lemma 4.5. $SKIP_a \circ PES_{ab} = PES_{ab} \circ SKIP_a = 1H_b$

Proof. Note that $b^!, b^!b^? \in (SKIP_a \,\mathbf{w}\, PES_{ab}) \upharpoonright \mathcal{A}^{Ext}_{SKIP_a, PES_{ab}}$. It follows by completeness (axiom 3 of Definition 3.4) that $b^!, b^!b^? \in SKIP_a \circ PES_{ab}$. The question now is whether $\epsilon \in SKIP_a \circ PES_{ab}$. To answer it, note that:

$$\begin{aligned} SKIP_a \circ (PES_{ab} \circ 1C_{cb}) &= SKIP_a \circ (CHAOS_a \circ 1H_c) && \text{(Fact 5)} \\ &= (SKIP_a \circ CHAOS_a) \circ 1H_c && \text{(assoc. - Def. 3.4(1))} \\ &= STOP \circ 1H_c = 1H_c && \text{(Fact 4, Fact 1)} \end{aligned}$$

First suppose that $\epsilon \in SKIP_a \circ PES_{ab}$, which implies $SKIP_a \circ PES_{ab} = 1NH_b$. Then $(SKIP_a \circ PES_{ab}) \circ 1C_{cb} = 1NH_b \circ 1C_{cb} = 1NH_c$, where the last equivalence is because $1C_{cb}$ does not guarantee that it will output on c. However, this breaks associativity of composition (axiom 1 of Definition 3.4):

$$(SKIP_a \circ PES_{ab}) \circ 1C_{cb} = 1NH_c \neq 1H_c = SKIP_a \circ (PES_{ab} \circ 1C_{cb})$$

Then suppose $\epsilon \notin SKIP_a \circ PES_{ab}$, which implies $SKIP_a \circ PES_{ab} = 1H_b$. Now we have $(SKIP_a \circ PES_{ab}) \circ 1C_{cb} = 1H_b \circ 1C_{cb} = 1H_c$, where the last equivalence is because $1C_{cb}$ always outputs on c once it has received an input on b. Then:

$$(SKIP_a \circ PES_{ab}) \circ 1C_{cb} = 1H_c = SKIP_a \circ (PES_{ab} \circ 1C_{cb})$$

Thus we have $SKIP_a \circ PES_{ab} = 1H_b$. By a symmetric reasoning and Lemma 4.1, we also have $PES_{ab} \circ SKIP_a = 1H_b$. □

We are now ready to show that $\circ$ does not exist.

Theorem 4.1 (impossibility). *Let* $- \circ - : \mathsf{HsLng} \times \mathsf{HsLng} \to \mathsf{HsLng}$. *Then* $\circ$ *cannot satisfy the three axioms given in Definition 3.4.*

Proof. By contradiction, assume that $\circ$ satisfies all the axioms. Let $\sigma = SKIP_a$, $\tau = EC_{abc}$ and $\rho = SKIP_b \circ 1F_{cd}$. With the results that we have established, we are able to determine the composition of σ, τ and ρ in both directions. We now show that we obtain one language in one direction and a different one in the other. First direction:

$$\begin{aligned}
\sigma \circ (\tau \circ \rho) &= \sigma \circ (EC_{abc} \circ (SKIP_b \circ 1F_{cd})) && \text{(substitution)}\\
&= \sigma \circ ((EC_{abc} \circ SKIP_b) \circ 1F_{cd}) && \text{(assoc. - Def. 3.4(1))}\\
&= \sigma \circ ((SKIP_b \circ EC_{abc}) \circ 1F_{cd}) && \text{(Lemma 4.1)}\\
&= \sigma \circ (CES_{ac} \circ 1F_{cd}) && \text{(Fact 7)}\\
&= \sigma \circ CES_{ad} && \text{(Fact 8)}\\
&= SKIP_a \circ CES_{ad} = 1NH_d && \text{(subst., Lemma 4.4)}
\end{aligned}$$

Second direction:

$$\begin{aligned}
(\sigma \circ \tau) \circ \rho &= (SKIP_a \circ EC_{abc}) \circ \tau && \text{(substitution)}\\
&= PES_{bc} \circ \tau && \text{(Fact 6)}\\
&= (PES_{bc} \circ SKIP_b) \circ 1F_{cd} && \text{(subst., assoc. - Def. 3.4(1))}\\
&= 1H_c \circ 1F_{cd} = 1H_d && \text{(Lemma 4.5)}
\end{aligned}$$

where the last equivalence is because $1F_{cd}$ forwards onto d the request $1H_c$ sends on c. To resume, we have:

$$\sigma \circ (\tau \circ \rho) = 1NH_d \neq 1H_d = (\sigma \circ \tau) \circ \rho$$

Then $\circ$ does not respect associativity (axiom 1 of Def. 3.4). □

5 Discussion

In this paper we have considered a specific protocol of asynchronous communication, the handshake protocol. We have defined handshake languages, by exploiting the standard formalisation of its properties in sets of traces [17]. Then we introduced a set of axioms for language composition and we showed that they may not all hold simultaneously.

We observe that:

1. our axioms do not refer to specific features of the handshake protocol, they refer to the extensional behaviour of composition in general, by imposing the loosest possible boundaries on what must and what may be observed;
2. the specificity of our class of languages HsLng implies that all the languages we used in the impossibility proof are contained in any larger class of languages as well.

The above two observations underpin the realisation that our impossibility proof has implications on a vast range of trace-models, not limited to models of the specific protocol considered in this study. One straightforward way to extend our result is by allowing messages to carry data (as in the original formalisation of the handshake protocol [17]). We

can even go one step further and allow dynamic creation of connections through name-passing (in π-calculus style [14]).

On the other hand, we can also extend the class of languages (and thus our result) by relaxing the constraints that define the handshake protocol, e.g. strict alternation between requests and acknowledges on each channel of communication. For instance, it is well-known that handshake languages can be encoded into the larger class of delay-insensitive languages [2, 12, 16], which also ensure the absence of computation-interference and of transmission-interference (mentioned in § 1).

To be clear, it is not our intention here to dismiss the whole literature of concurrent and non-deterministic trace-models as flawed, since it would be out of the scope of this paper. Many models achieve composition by sacrificing the information on quiescence (e.g. by considering prefix-closed languages), thus failing to represent deadlocks. Other models have different and often more elaborate ways of characterising deadlocks (e.g. failures and divergences models [1, 8]). Rather than rushing to dismiss all the neighbouring literature, we set it as the next challenge to check if and how our impossibility proof can be applied to variant trace-models, starting with models of delay-insensitive and *speed-independent* communications [10, 11] (the latter being a yet larger class).

However, we can say that characterising divergences simply as infinite chatterings (as we did), without further "decorating" languages with special traces, is arguably the most natural way of proceeding, without altering the nature of the model. Then it may be unsurprising that our proof reflects an observation that has kept two different "schools" debating for some time: the school of linear-time (e.g. [7, 15]) and that of branching-time (e.g. [13, 14, 18]) semantics, where trace-semantics belong to the former. The observation is that trace models do not allow to distinguish the following two CCS processes: $a.b + a.c$ and $a.(b + c)$; while composing them with $\bar{a}.\bar{b}$ may give rise to a deadlock in one case but not in the other. Nonetheless, as our proof shows, the absence of a satisfactory notion of composition in trace-models is not just a mere consequence of such well-known observation: it does require a careful selection of a set of ad-hoc languages to exploit such behavioural understanding in a formal proof.

Another objection one can make to our proof is that Lemma 4.2 was proved only because our languages are non-empty by definition. One can argue that the language on the empty alphabet should be $\emptyset$ (the empty language), rather than $\{\epsilon\}$ (the language containing only the empty trace). We note that, from an observational viewpoint, $\emptyset$ and $\{\epsilon\}$ are indistinguishable, which is the reason why we did not allow handshake languages to include $\emptyset$. Moreover, adding the empty language does not remedy the problem with composition, as we show in § A.

A possibility we have not considered is to allow languages to contain infinite traces. However, this may not be a viable alternative, since the three main properties stated in Definition 2.4 each enforce a closedness condition which would be undecidable over sets of infinite traces. Note also that in such a framework, the problem would not be that of finding a good notion of composition, since infinite chatterings would be explicitly represented through interleavings. The aim there would be to give criteria to characterise the infinite traces that a language should contain. Our guess is that it should not be possible either, although we leave it as a future work.

Another question we leave open for now is that of finding the minimal limitations to non-determinism and/or concurrency, which allow a satisfactory notion of composition.

References

1. S. Brooks and A. Roscoe. An improved failures model for communicating processes. *Lecture Notes in Computer Science*, 197:281–305, 1985.
2. W. Clark and C. Molnar. Macromodular computer systems. In *Computers in Biomedical Research*, volume IV, chapter 3. Academic Press, 1974.
3. L. Fossati. Handshake games. In *Developments in Computational Models, DCM'06*, volume 171 of *ENTCS*, pages 21–41. Elsevier, 2007.
4. L. Fossati. *Modeling the Handshake Protocol for Asynchrony*. PhD thesis, Università di Torino & Université 'Denis Diderot' Paris VII, Turin, 2009.
5. L. Fossati and D. Varacca. A calculus for handshake configurations. In *Foundations of Software Science and Computation Structures, FoSSaCS'09*, volume 5504 of *LNCS*, pages 227–241. Springer, 2009.
6. L. Fossati and D. Varacca. A petri net model of handshake protocols. In *Interaction and Concurrency Experience, ICE'08*, volume 229 of *ENTCS*, pages 59–76. Elsevier, 2009.
7. C. Hoare. A model for communicating sequential processes. In *On the Construction of Programs*. Cambridge University Press, 1980.
8. C. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.
9. http://www.keil.com/dd/docs/datashts/handshake/ht80c51.pdf.
10. M. B. Josephs. Receptive process theory. *Acta Informatica*, 29(1):17–31, 1992.
11. M. B. Josephs and J. T. Udding. Delay-insensitive circuits: an algebraic approach to their design. *Lecture Notes in Computer Science*, 458:342–366, 1990.
12. Macromodules Project. Macromodular computer systems. Technical Report 4, Computer Systems Laboratory, Washington University, 1967.
13. R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer, 1980.
14. R. Milner, G. Parrow, and D. Walker. A calculus of mobile processes. *Information and Computation*, 100(1):1–77, 1992.
15. S. Nain and M. Y. Vardi. Trace semantics is fully abstract. In *LICS*, pages 59–68, 2009.
16. J. Udding. *Classification and Composition of Delay-Insensitive Circuits*. PhD thesis, Eindhoven University of Technology, Eindhoven, 1984.
17. K. Van Berkel. *Handshake Circuits: an Asynchronous Architecture for VLSI Design*, volume 5 of *Cambridge International Series on Parallel Computation*. Cambridge University Press, 1993.
18. R. Van Glabbeek. The linear time - branching time spectrum I: the semantics of concrete sequential processes. *Handbook of Process Algebra*, pages 3–99, 2001.

A Taking Empty Languages into Account

In the proof of Lemma 4.2, we were able to determine the result of composing a pair of infinite languages only because our languages are non-empty by definition. However, one could argue that the language on the empty alphabet should be $\emptyset$ (the empty language), rather than $\{\epsilon\}$ (the language containing only the empty trace). We note that, from the point of view of an external observer, $\emptyset$ and $\{\epsilon\}$ are indistinguishable, which is the reason why we did not allow handshake languages to include $\emptyset$.

Nonetheless, let $EMPTY = \emptyset$ denote the empty language and let us consider an alternative set of handshake languages $\mathsf{HsLng} \cup \{EMPTY\}$, just to show that this does not help finding a satisfactory notion of composition. In Definition 2.4 we avoided any explicit reference to the alphabet on which a language is defined. However, in order to extend receptiveness (Condition 3 in Def. 2.4) we ask that $EMPTY$ may not be defined on an alphabet containing negative ports. The principle is the usual one for receptiveness: a language should be ready to accept any input from the environment. Formally:

Definition A.1 ($\emptyset$-receptive, $\mathsf{HsLng}^{\emptyset}$). Let $\sigma \in \mathsf{HsLng} \cup \{EMPTY\}$ be a language on alphabet $\mathcal{A}$. We say that σ is *$\emptyset$-receptive* iff, for any $a \in ch(\mathcal{A})$, $\langle a, - \rangle \in \mathcal{A}$ implies $\sigma \neq EMPTY$. We write $\mathsf{HsLng}^{\emptyset}$ to denote the set of languages in $\mathsf{HsLng} \cup \{EMPTY\}$ which additionally satisfy the $\emptyset$-receptiveness condition.

In the proof, we also use the following additional axiom, asking that composition be closed under channel renaming. Below, let $\sigma[b/a]$ be the language obtained from σ after replacing all occurrences of $a^!$ and of $a^?$ by occurrences of $b^!$ and of $b^?$, respectively.

Definition A.2 (renaming). A composition $- \circ - : \mathsf{HsLng}^{\emptyset} \times \mathsf{HsLng}^{\emptyset} \longrightarrow \mathsf{HsLng}^{\emptyset}$ is *closed under renaming* if for all $b \notin ch(\mathcal{A}_\sigma) \cup ch(\mathcal{A}_\tau)$ and for all $a \in ch(\mathcal{A}_\sigma) \cup ch(\mathcal{A}_\tau)$, $\sigma[b/a] \circ \tau[b/a] = (\sigma \circ \tau)[b/a]$.

To resume, a satisfactory notion of composition should satisfy the following:

Definition A.3 ($\mathsf{HsLng}^{\emptyset}$-axioms). In $\mathsf{HsLng}^{\emptyset}$, a *satisfactory notion of composition* $- \circ -$: $\mathsf{HsLng}^{\emptyset} \times \mathsf{HsLng}^{\emptyset} \to \mathsf{HsLng}^{\emptyset}$ is required to be closed under renaming (Definition A.2), associative (axiom 1 from Definition 3.4) and such that soundness and completeness (axioms 2 and 3 from Definition 3.4) hold only for languages in HsLng.

The reason why we did not extend soundness and completeness to $EMPTY$ is because $EMPTY$ has empty weave and divergence set when composed with any other language.

The main issue is to understand how $EMPTY$ should compose with other languages and especially with non-trivial ones, where a language $\sigma \in \mathsf{HsLng}^{\emptyset}$ is *non-trivial* if $\sigma \notin \{EMPTY,\ STOP\ \}$. First we show that it may only behave in two possible ways.

Lemma A.1. *Let σ be a non-trivial language. Then:*

$$\begin{array}{c} (\sigma \circ EMPTY = \sigma \ \vee \ \sigma \circ EMPTY = EMPTY) \\ \wedge \\ (EMPTY \circ \sigma = \sigma \ \vee \ EMPTY \circ \sigma = EMPTY) \end{array}$$

Proof. Let $ch(\mathcal{A}_\sigma) = \{a_1, \dots a_n\}$ and let $\tau = \sigma[b_1/a_1] \dots [b_n/a_n]$, where $b_i \notin ch(\mathcal{A}_\sigma)$, for all $i \in \{1, \dots, n\}$.

By contradiction, suppose there is $s \in \sigma \circ EMPTY$ such that $s \notin \sigma$. Since σ is non-trivial, τ is also non-trivial, hence non-empty. Then take $s' \in \tau$ and let t be an interleaving of s and s'. Since $\mathcal{A}_{\sigma \circ EMPTY}$ and $\mathcal{A}_\tau$ are disjoint, $t \in (\sigma \circ EMPTY) \circ \tau$. On the other hand, $t \notin \sigma \circ (EMPTY \circ \tau)$, since $s \notin \sigma$. And this contradicts the associativity of $- \circ -$. Hence we have:

$$\sigma \circ EMPTY \subseteq \sigma$$

In the same way we could show that there may not exist $s \in \sigma$ such that $s \notin \sigma \circ EMPTY$. But note that we would need the additional assumption $\sigma \circ EMPTY \neq EMPTY$, which brings us to conclude that either $\sigma \circ EMPTY = \sigma$ or $\sigma \circ EMPTY = EMPTY$. We show $(EMPTY \circ \sigma = \sigma \vee EMPTY \circ \sigma = EMPTY)$ in the same way. □

Next we show that $EMPTY$ may not behave as the identity for $- \circ -$.

Lemma A.2. *There exists a non-trivial $\sigma \in \mathsf{HsLng}^\emptyset$ such that:*

$$\sigma \circ EMPTY \neq \sigma \neq EMPTY \circ \sigma$$

Proof. Note that if only $EMPTY \circ 1NH_b = 1NH_b$, the whole impossibility proof of § 4 would still hold even if we dismissed Lemma 4.2 to allow $SKIP_a \circ RUN_a = EMPTY$ (Lemma 4.2 is used only once in the proof, and that is within the proof of Lemma 4.3). Then it must be the case that $1NH_b \neq EMPTY \circ 1NH_b$. Note also that, by slightly remodelling the proof of § 4, we can similarly show $1NH_b \circ EMPTY \neq 1NH_b$. □

Similarly we exclude that $EMPTY \circ -$ and $- \circ EMPTY$ may act as the constant $EMPTY$.

Lemma A.3. *There exists a non-trivial $\sigma \in \mathsf{HsLng}^\emptyset$ such that:*

$$\sigma \circ EMPTY \neq EMPTY \neq EMPTY \circ \sigma$$

Proof. Just take as an example $SKIP_a$, although any handshake language defined on a port with negative polarity would do. Note that $\mathcal{A}_{SKIP_a} = \{\langle a, - \rangle\}$ consists of a port with negative polarity. W.l.o.g. assume the language $EMPTY$ occurring before the composition is defined on the empty alphabet, so that when we compose σ with $EMPTY$ (in any order) we obtain a language which is still defined on $\mathcal{A}_{SKIP_a}$. Hence the composition may not yield $EMPTY$, or $\emptyset$-receptiveness (Definition A.1) would break. □

Then we conclude:

Theorem A.1 ($\mathsf{HsLng}^\emptyset$-impossibility). *Let $- \circ - : \mathsf{HsLng}^\emptyset \times \mathsf{HsLng}^\emptyset \to \mathsf{HsLng}^\emptyset$. Then $\circ$ may not simultaneously satisfy all the axioms given in Definition A.3.*

Proof. By Lemma A.1, the result of composing $EMPTY$ with any non-trivial language is either $EMPTY$ or that language. Then by Lemma A.3, there is a non-trivial language σ such that $\sigma \circ EMPTY = \sigma$, and by Lemma A.2, there is a non-trivial language τ such

that $EMPTY \circ \tau = EMPTY$. Note also that renaming (Definition A.2) allows us to choose σ and τ such that $ch(\mathcal{A}_\sigma) \cap ch(\mathcal{A}_\tau) = \emptyset$. Then:

$$(\sigma \circ EMPTY) \circ \tau = \sigma \circ \tau$$

where $\sigma \circ \tau$ contains all the interleavings of traces of σ with traces of τ, since $ch(\mathcal{A}_\sigma) \cap ch(\mathcal{A}_\tau) = \emptyset$. Hence it is different from both σ and τ, since σ and τ are both non-trivial. On the other hand we have:

$$\sigma \circ (EMPTY \circ \tau) = \sigma \circ EMPTY = \sigma$$

which contradicts the associativity axiom requested by Definition A.3. □