The Calculus of Handshake Configurations

Luca Fossati[1,2]⋆ and Daniele Varacca[2]

[1] Dip. di Informatica - Università di Torino, Italia
[2] PPS - CNRS & Univ. Paris Diderot, France

Abstract. Handshake protocols are asynchronous protocols that enforce several properties such as absence of transmission interference and insensitivity from delays of propagation on wires. We propose a concurrent process calculus for handshake protocols . This calculus uses two mechanisms of synchronization: rendez-vous communication à la CCS, and shared resource usage. To enforce the handshake discipline, the calculus is endowed with a typing system .
We provide an LTS semantics of the calculus and show that typed processes denote handshake protocols. We give the calculus another semantics in terms of a special kind of Petri nets called handshake Petri nets. We show that this semantics is complete and fully abstract with respect to weak bisimilarity.

Key words: Handshake protocols, Petri nets, process calculus, types.

1 Introduction

Asynchronous circuits are used to design systems where the local activity of each subunit is not restrained by some global condition, like the long time intervals imposed by a system clock. When designing such systems, one has to face several questions. How do we know when a message we sent has reached its destination so that we can use the same channel again, i.e. how can we avoid *transmission interference*? How can we ensure the correct behavior regardless of computational speeds of single modules and propagation delays over wires, i.e. how can we enforce *delay-insensitivity*?

Handshake protocols try to answer these questions by imposing an interactive communication discipline. In particular, the protocols require that after a circuit has sent a message on a channel, it has to wait for a confirmation that the message was received before sending again on the same channel. This requirement alone is enough to rule out transmission interference. For their simplicity and efficiency, handshake protocols have been employed by enterprises like Philips and Sun in the development of a series of VLSI chips [9].

The first attempts to formalize delay-insensitive protocols and their properties employed trace sets [17]. In particular, the first model to specifically address the handshake case was given in [18]. Trace models have been able to neatly formalize the properties of handshake protocols which ensure delay-insensitivity, but so far they have failed in representing correctly their composition [4].

To overcome this limitation, we propose an alternative approach to modeling handshake protocols: we propose a process calculus inspired by Robin Milner's Calculus of

⋆ Corresponding author: fossati@di.unito.it

Communicating Systems (CCS) [13]. Similarly to CCS, our calculus defines concurrent processes that communicate via rendez-vous channels. However, in order to ensure the handshake discipline, the calculus features another synchronization mechanism, by means of shared resources, reminiscent of coordination languages like Linda [7]. Also, the calculus is endowed with a linear typing system, inspired by [11, 19]. These design choices allow to express the external behavior of a handshake protocol, along with a more complex internal behavior.

We say that this is the first independent syntactical description of the handshake behavior, as our *handshake configurations* are independent from any semantical interpretation while the handshake behavior is ensured by the typing system. This was not the case in previous process algebras for handshake protocols [18, 10], where only processes whose trace semantics satified the handshake behavior were considered, thus processes were trace sets and inevitably suffered from the compositionality problems observed in the underlying trace model.

We then compare our calculus and the trace model by defining, for each configuration, the corresponding set of *quiescent* traces, i.e. the traces corresponding to computations that may not be extended with an output event. We show that this quiescent trace semantics is sound w.r.t. Van Berkel's definition [18]. By means of an example, we show that quiescent trace equivalence is not a congruence w.r.t. parallel composition. This confirms the intuition that trace models of handshake protocols are not informative enough, and their branching structure needs to be taken into account. Indeed weak bisimilarity is a congruence for our calculus.

To show the expressive power of the calculus, we give it a Petri net semantics, where the handshake discipline is imposed by restricting a net's external structure. We studied this model in details in a previous work [6], where we showed that it captures precisely the behavior of a protocol, in the sense that there is a protocol for each net and a net (possibly with an infinite number of places and/or transitions) for each protocol. In this work we show that there is a correspondence between handshake processes and *finite* Petri nets, in the sense that for each finite handshake Petri net, there is a weakly bisimilar process.

The graphical approach to the formal analysis of asynchrony is not a new one ([1],[12],...). In particular Dan Ghica developed a language for asynchronous hardware design by taking inspiration from the Geometry of Interaction and handshake circuits [8]. However his goal was to improve previous hardware design languages [18, 3] and not to capture all handshake behaviors.

The paper is structured as follows. In Section 2 we recall the first formalization of handshake protocols as we introduce the notion of handshake language. In Section 3, we present syntax, operational semantics and type system of our calculus. We show that the set of quiescent traces of a typed configuration is a handshake language. We show that weak bisimilarity is a congruence, while quiescent trace semantics is not. In Section 4, we present handshake Petri nets, with some examples to show how they work. In Section 5, we present the interpretation of the calculus into handshake nets, and we show that it is fully abstract with respect to weak bisimilarity. To conclude, we show the universality of the semantics, by showing that every bisimilarity class of finite handshake nets is denoted by a process.

2 Handshake protocols

In this section we recall the background properties of handshake protocols and introduce the notion of *handshake language*.

Definition 1. *A* handshake structure *is a pair* $\langle Ports, d\rangle$*, where* $Ports$ *is a finite set of* ports *and the function* $d : Ports \to \{!, ?\}$ *determines a direction for each port,* active *or* passive.

As we shall see, active ports are allowed to start a communication, while passive ports are initially waiting. For the rest of this section let $\langle Ports, d\rangle$ be a handshake structure. For each port a, there are two possible messages: a (input message), and $\bar{a}$ (output message). Let t be a finite trace on the alphabet of messages $\cup_{a\in Ports}\{a, \bar{a}\}$. t is a *handshake trace* on $\langle Ports, d\rangle$ if for all $a \in Ports$:

– $t \upharpoonright \{a, \bar{a}\} = \bar{a}a\bar{a}a \ldots$ when $d(a) =!$;
– $t \upharpoonright \{a, \bar{a}\} = a\bar{a}a\bar{a} \ldots$ when $d(a) =?$;

Given a set of traces S we write $S^{\leq}$ for its prefix-closure. Let σ be a set of handshake traces, $s \in \sigma^{\leq}$ is *passive* in σ if and only if there is no extension of s in σ obtained by appending an output message: $\forall s \cdot m \in \sigma^{\leq}$, m is an input message. We write $Pas(\sigma)$ for the set of passive traces in $\sigma^{\leq}$.

We define $\mathbf{r}_{Ports}$ as the smallest binary relation which is closed by reflexivity, transitivity and concatenation and such that for any distinct ports $a, b \in Ports$:

1. $a\bar{b}\ \mathbf{r}_{Ports}\ \bar{b}a$;
2. $\bar{a}b\ \mathbf{r}_{Ports}\ b\bar{a}$;
3. $ab\ \mathbf{r}_{Ports}\ ba$

We say that s *reorders* t in $Ports$ if $s\ \mathbf{r}_{Ports}\ t$.

Let s be a handshake trace and $a \in Ports$. We write $a\ \mathbf{x}_{Ports}\ s$ if sa is still a handshake trace. Finally:

Definition 2. *A* handshake language σ *on* $\langle Ports, d\rangle$ *is a non-empty set of finite handshake traces on* $\langle Ports, d\rangle$ *such that:*

1. $Pas(\sigma) \subseteq \sigma$ *(closed under passive prefixes);*
2. $(t \in \sigma \wedge s\ \mathbf{r}_{Ports}\ t) \Rightarrow s \in \sigma$ *(reorder closed);*
3. $(s \in \sigma^{\leq} \wedge a\ \mathbf{x}_{Ports}\ s) \Rightarrow s \cdot a \in \sigma^{\leq}$ *(receptive).*

Closedness under passive prefixes, rather than under any prefix as it is usually the case for trace semantics, allows us to represent *may&must nondeterminism*. We want to represent systems which are able not only to make an exclusive choice between two outputs, but also to choose between sending an output and not doing anything.

The intuition is thus that a trace in a handshake language represents a *quiescent* execution of a protocol, that is an execution that ends in a state in which the system may decide to wait for more inputs before sending any output. By definition, after a passive trace the system cannot do anything but receiving. Then all passive traces correspond to quiescent executions.

Reorder-closedness says that a message m' cannot "block" a message m'' on a different channel unless m' is an input and m'' an output. The intuition is that inputs may carry necessary information and thus may block, while the transmission of an output may require informations and can thus be blocked.

Finally, receptiveness means that whenever it is the environment's turn to send a message, the system must be ready to accept it.

Definition 2 is like VanBerkel's original definition of handshake process [18] without data-passing. No satisfactory definition of composition for handshake languages exists. In particular the definition given by VanBerkel is not associative, as shown by the first author [4] using a counter-example by Roscoe [16].

In the following sections, handshake languages will be used as a yardstick against which to measure the correctness of other descriptions of handshake protocols.

3 The calculus

In this section we provide the formal definition for our *Calculus of Handshake Configurations (CHC)*. We stress that we do not model data-passing as we are only interested in the communication protocol. The calculus is endowed with two communication mechanisms. Besides the external communication via rendez-vous channels, there is also a form of internal communication, invisible to the outside, where actions may require resources in order to be performed and may release resources for other actions to use. This is necessary to model internal synchronizations between different ports of the same system. However, different systems shall communicate only through channels.

3.1 Syntax and Operational Semantics

We consider a set A of *channels* denoted by a, b[3], and a set R of *resources* denoted by r, k. The syntax of the calculus uses three syntactic categories: *threads*, *processes* and *handshake configurations*. Threads are purely sequential and allow prefixing while processes are parallel compositions of threads. The prefixes are *input* and *output* actions on a finite set of resources. As we will see later, input actions *release* resources and output actions *use* resources. Let $\varDelta \subseteq A$:

$$\begin{array}{rl}
act ::= a^{\{r_1,\dots,r_n\}} \mid \overline{a}_{\{r_1,\dots,r_n\}} & \text{Actions}\\
T ::= \mathbf{0} \mid act.T \mid \mathbf{Rec}\ T & \text{Threads}\\
P, Q ::= T \mid P \mid Q \mid P \setminus \varDelta & \text{Processes}
\end{array}$$

A handshake configuration is composed of a process P along with a *multiset* of resources S for internal synchronization. A configuration can be *open* or *closed*:

$$M ::= \wr P, S \wr \mid \langle P, S \rangle \quad \text{Open and closed configurations}$$

Intuitively, open configurations represent systems under construction, whose resources are still accessible to the environment. Closed configurations represent completed systems and can only communicate via handshake channels.

[3] We will use channels to model ports, but we prefer to keep the conceptual difference between the two notions

$$\frac{}{\wr a^{\{r_1,\dots,r_n\}}.T, S \wr \xrightarrow{a} \wr T, S + \{r_1,\dots,r_n\} \wr}\ \text{(inev)} \qquad \frac{\wr T \cdot \mathbf{Rec}\ T, S \wr \xrightarrow{e} \wr T', S' \wr}{\wr \mathbf{Rec}\ T, S \wr \xrightarrow{e} \wr T', S' \wr}\ \text{(rec)}$$

$$\frac{}{\wr \overline{a}_{\{r_1,\dots,r_n\}}.T, S + \{r_1,\dots,r_n\} \wr \xrightarrow{\overline{a}} \wr T, S \wr}\ \text{(outev)} \qquad \frac{M \xrightarrow{e} M' \quad ch(e) \notin \Delta}{M \setminus \Delta \xrightarrow{e} M' \setminus \Delta}\ \text{(res)}$$

$$\frac{M \xrightarrow{e} M'}{M \mid N \xrightarrow{e} M' \mid N}\ \text{(par1)} \qquad \frac{M \xrightarrow{\overline{a}} M' \qquad N \xrightarrow{a} N'}{M \mid N \xrightarrow{\tau} M' \mid N'}\ \text{(par2)}$$

$$\frac{P \equiv P' \quad \wr P', S \wr \xrightarrow{e} \wr Q', S' \wr \quad Q \equiv Q'}{\wr P, S \wr \xrightarrow{e} \wr Q, S' \wr}\ \text{(struct)} \qquad \frac{\wr P, S \wr \xrightarrow{e} \wr Q, S' \wr}{\langle P, S \rangle \xrightarrow{e} \langle Q, S' \rangle}\ \text{(closure)}$$

Fig. 1. Labeled transition semantics

The operational semantics is given in terms of an LTS over handshake configurations. Labels are channels with their polarity, plus the unobservable label:

$$e ::= \overline{a} \mid a \mid \tau \qquad a \in A$$

Given an observable label, the function ch returns the channel on which it occurred. Formally: $ch(\overline{a}) = ch(a) = a$ for any channel a. The definition of the operational semantics is simplified thanks to the congruence ($\equiv$) between processes:

$$P \mid Q \equiv Q \mid P \qquad \mathbf{Rec}\ \mathbf{0} \equiv \mathbf{0}$$

Let $res(P)$ be the set of resources of a process P. As meta-notation, we define sequential composition of threads $T \cdot T'$:

$$(act.T) \cdot T' = act.(T \cdot T') \qquad T \cdot T' = T' \ \ (T \equiv \mathbf{0}) \qquad (\mathbf{Rec}\ T) \cdot T' = \mathbf{Rec}\ T \ \ (T \not\equiv \mathbf{0})$$

and we extend process operators to configurations:

$$\wr P_1, S_1 \wr \mid \wr P_2, S_2 \wr = \wr P_1 \mid P_2, S_1 + S_2 \wr \qquad \wr P, S \wr \setminus \Delta = \wr P \setminus \Delta, S \wr$$

$$\langle P_1, S_1 \rangle \mid \langle P_2, S_2 \rangle = \langle \mu_1(P_1) \mid \mu_2(P_2), \mu_1(S_1) + \mu_2(S_2) \rangle \qquad \langle P, S \rangle \setminus \Delta = \langle P \setminus \Delta, S \rangle$$

where + denotes the union of multisets and $\mu_1 : res(P_1) \cup S_1 \to R_1$ and $\mu_2 : res(P_2) \cup S_2 \to R_2$ are injective functions between resources such that R_1 and R_2 are disjoint and all the resources they contain are fresh. Moreover $\mu(S)$ is the point-to-point application of the function μ to the multiset S, while $\mu(P)$ is the process obtained from P by renaming any label occurrence according to μ. This guarantees that two closed configurations can only communicate via channels .

Note that we do not define the parallel composition of an open and a closed configuration. The idea is that we may combine two different parts of a system (in the construction stage) or two completed systems (for interaction), but we may not combine a system under construction with a completed one.

$$\frac{a \in A}{\mathbf{0} \rhd !a}\ \text{(ax)} \qquad\qquad \frac{T \rhd !a}{\mathbf{Rec}\, T \rhd !a}\ \text{(rec)}$$

$$\frac{T \rhd ?a}{\overline{a}_{\{r_1,\dots r_n\}}.T \rhd !a}\ \text{(outpref)} \qquad\qquad \frac{T \rhd !a}{a^{\{r_1,\dots r_n\}}.T \rhd ?a}\ \text{(inpref)}$$

$$\frac{P \rhd \Gamma' \qquad Q \rhd \Gamma'' \qquad \forall a \in Dom(\Gamma') \cap Dom(\Gamma''),\ \Gamma'(a) \neq \Gamma''(a)}{(P \mid Q) \setminus (Dom(\Gamma') \cap Dom(\Gamma'')) \rhd \Gamma' \odot \Gamma''}\ \text{(par)}$$

$$\frac{P \rhd \Gamma}{\wr P, S \wr \rhd \Gamma}\ \text{(oconf)} \qquad\qquad \frac{P \rhd \Gamma}{\langle P, S \rangle \rhd \Gamma}\ \text{(cconf)}$$

Fig. 2. Handshake types

The derivation rules for the operational semantics are shown in Figure 1. When an input occurs, a set of resources becomes available; while an output requires a set of resources in order to occur, then the used resources disappear. The other rules are quite standard. Note however that the operational distinction between open and closed configurations comes from the two distinct cases of composition given above. In the parallel composition of open configurations, one side may influence the other by modifying a shared resource, as no renaming takes place. This is not possible for closed configurations, as renaming prevents the sharing of resources.

A sequence of transitions $M_0 \xrightarrow{e_0} M_1 \dots M_n \xrightarrow{e_n} M$ is denoted $M_0 \overset{t}{\twoheadrightarrow} M$, where $t = e_0 \dots e_n$. The string t is called the *strong* trace of the sequence, while the *weak* trace is the restriction of t to the labels other than τ. Strong ($\sim$) and weak ($\approx$) bisimilarity are also defined as usual [13] on the labeled transition system for CHC.

3.2 Typing System

A *type* Γ is a partial function from channel names to $\{!, ?\}$. We will use the shorthand notation $!a$ or $?a$ to describe a type defined on channel a, and commas to join types. We say that a is *active* in Γ when $\Gamma(a) =!$ and we say it is *passive* when $\Gamma(a) =?$.

Let Γ' and Γ'' be two types and let a be a channel. Let us define the function $\Gamma' \odot \Gamma'' : (Dom(\Gamma') \setminus Dom(\Gamma'')) \cup (Dom(\Gamma'') \setminus Dom(\Gamma')) \to \{!, ?\}$ such that:

– $\Gamma' \odot \Gamma''(a) = \Gamma'(a)$, when $a \in Dom(\Gamma') \setminus Dom(\Gamma'')$;
– $\Gamma' \odot \Gamma''(a) = \Gamma''(a)$, when $a \in Dom(\Gamma'') \setminus Dom(\Gamma')$.

Typing judgements are of the form $T \rhd \Gamma,\ P \rhd \Gamma,\ M \rhd \Gamma$, where T is a thread, P a process, M a configuration and Γ a type. The typing rules are shown in Figure 2. The empty thread is active: this models receptiveness, because a thread of passive type must always be able to perform another input. The following three rules guarantee that threads are alternating on each channel. The parallel composition of two processes is allowed only if threads on the same channel have dual types. These channels must then be restricted so that no other process can communicate on them. This models the point-to-point communication discipline of handshake protocols. Note that resources do not play any role in the typing.

The following results show the intuition behind the typing system.

Lemma 3 (Reduction). *Let M be a configuration such that $M \rhd \Gamma$. Then:*

- $M \xrightarrow{a} \quad \Longleftrightarrow \quad \Gamma(a) = ?$;
- $M \xrightarrow{\overline{a}} \quad \Rightarrow \quad \Gamma(a) = !$;
- $M \xrightarrow{e} M' \wedge e \neq \tau \quad \Rightarrow \quad M' \rhd \Gamma'$ *s.t.* $Dom(\Gamma') = Dom(\Gamma) \wedge \forall b \in Dom(\Gamma), b \neq ch(e) \leftrightarrow \Gamma(b) = \Gamma'(b)$;
- $M \xrightarrow{\tau} M' \quad \Rightarrow \quad M' \rhd \Gamma$;

Corollary 4 (Subject Reduction). *Let $M \rhd \Gamma$ and $M \overset{s}{\twoheadrightarrow} M'$ then there is a type Γ' such that $M' \rhd \Gamma'$.*

3.3 Examples

As a first example, we show a configuration representing the *OR* handshake protocol.

$$OR = \langle a^{\{r_1\}}.\mathbf{Rec}\ \overline{a}_{\{r_2\}}.a^{\{r_1\}}.\mathbf{0} \mid \mathbf{Rec}\ \overline{b}_{\{r_1\}}.b^{\{r_2\}}.\mathbf{0} \mid \mathbf{Rec}\ \overline{c}_{\{r_1\}}.c^{\{r_2\}}.\mathbf{0}, \emptyset\rangle$$

We have that $OR \rhd ?a, !b, !c$. When a request on the passive port a arrives, the resource r_1 becomes available and this enables *OR* to send a request on either active port b, c. An acknowledge to this last request enables an acknowledge to the first one. The second configuration represents the *MIX* protocol.

$$MIX = \langle b^{\{k_1\}}.\mathbf{Rec}\ \overline{b}_{\{k_2\}}.b^{\{k_1\}}.\mathbf{0} \mid c^{\{k_1\}}.\mathbf{Rec}\ \overline{c}_{\{k_2\}}.c^{\{k_1\}}.\mathbf{0} \mid \mathbf{Rec}\ \overline{d}_{\{k_1\}}.d^{\{k_2\}}.\mathbf{0}, \emptyset\rangle$$

We have that $MIX \rhd ?b, ?c, !d$. Each time an environment request arrives (on either passive port b, c), the component *MIX* handshakes on its active port d and after completion it acknowledges to the first request. If, by the time the handshake on the active port is complete, the environment has sent a request on the other port, *MIX* chooses nondeterministically which request to acknowledge first.

The two protocols can be composed in parallel, communicating on the common ports. We have $(OR \mid MIX) \setminus \{b, c\} \rhd ?a, !d$.

3.4 Soundness

In this section we show that typed CHC configurations indeed define handshake languages, using the weak traces of the labeled transition semantics.

Each feature of the calculus plays a role in modeling the handshake discipline. Let us see informally how. First of all, handshake languages alternate input and output on the same port. This is enforced by the typing system. The reorder closure is guaranteed by the fact that different ports are on different parallel threads. The only reordering that is in general not allowed is when an input blocks an output. An input can block an output because an output may need resources that will only be provided by the input. Finally, receptiveness is guaranteed by the fact that inputs do not need resources, and can always occur, provided that the alternation with the corresponding outputs is respected.

In order to denote handshake languages we consider the weak traces of the transition sequences of a configuration. If we considered the traces of all transition sequences, the denoted languages would always be prefix closed and some handshake languages would excape us. To characterize the larger class of languages closed under passive prefixes, we consider only the traces of the *quiescent* transition sequences.

A configuration M is *quiescent* if it cannot (weakly) perform an output, i.e. if there is no transition sequence of the form $M \overset{(\tau)^*}{\twoheadrightarrow} \xrightarrow{\bar{a}}$, for any channel a. A transition sequence $M \overset{t}{\twoheadrightarrow} M'$ is *quiescent* if M' is.

Definition 5. *Let M be a handshake configuration. We define $HL(M)$ to be the set of weak traces of all the quiescent transition sequences which start from M.*

Let M be a configuration and Γ a type such that $M \rhd \Gamma$. The handshake structure $HS(\Gamma) = \langle Ports_\Gamma, d_\Gamma \rangle$ is defined by setting $Ports_\Gamma = Dom(\Gamma)$ and $d_\Gamma = \Gamma$.

Proposition 6 (Soundness). *Let M be a handshake configuration, such that $M \rhd \Gamma$. Then $HL(M)$ is a handshake language on the handshake structure $HS(\Gamma)$.*

We observe, however, that the other direction, the fact that each language is the denotation of some configuration, cannot be established. This is due to the presence of non recursive handshake languages which could never be captured by finite configurations. It would still be interesting to characterize the class of handshake languages that correspond to CHC configurations. We leave this as future work.

3.5 Compositionality

Open configurations can communicate via shared resources, but this is not directly observable in the labeled transition semantics. Thus we cannot expect a labeled equivalence to be fully congruent for them. However weak bisimilarity is a congruence with respect to composition of closed configurations:

Proposition 7. *Let M_1, M_2, N be closed handshake configurations such that $M_1, M_2 \rhd \Gamma$, $N \rhd \Gamma'$ and $(M_1 \mid N) \setminus \Delta \rhd \Gamma \odot \Gamma'$, $(M_2 \mid N) \setminus \Delta \rhd \Gamma \odot \Gamma'$, where $\Delta = Dom(\Gamma) \cap Dom(\Gamma')$. Then $M_1 \approx M_2$ implies $(M_1 \mid N) \setminus \Delta \approx (M_2 \mid N) \setminus \Delta$.*

This is consistent with our intepretation of resources as *internal* means of communication. Our main goal was to describe the *externally observable* behavior of a system and we do so by considering only those configurations whose resources cannot be accessed by the environment.

In Section 2 we talked about the difficulty of finding a good definition of composition for handshake languages. This intuition is confirmed as “quiescent trace equivalence” is not a congruence. Consider the following processes:

$$P_1 = \bar{c}_{\{r_1,r_2\}}.c^{\{\}}.\mathbf{0} \mid \bar{b}_{\{r_3,r_1\}}.b^{\{r_1\}}.\mathbf{Rec}\ \bar{b}_{\{r_1\}}.b^{\{r_1\}}.\mathbf{0} \mid (\bar{d}_{\{r_3,r_2\}}.d^{\{r_3\}}.\mathbf{0} \mid d^{\{\}}.\bar{d}_{\{\}}.d^{\{\}}.\mathbf{0}) \setminus \{d\}$$

$$P_2 = \bar{c}_{\{r_1\}}.c^{\{\}}.\mathbf{0} \mid \mathbf{Rec}\ \bar{b}_{\{r_1\}}.b^{\{r_1\}}.\mathbf{0}\ \ .$$

Consider the closed configurations $M_1 = \langle P_1, \{r_1, r_2, r_3\}\rangle$ and $M_2 = \langle P_2, \{r_1, r_2, r_3\}\rangle$. They are both interpreted as the same handshake language:

$$HL(M_1) = \{\overline{c}, \overline{c}c, \overline{b}, \overline{b}b\overline{c}, \overline{b}b\overline{c}c, \overline{b}b\overline{b}, \ldots\} = HL(M_2)$$

however, if we compose them with $N = \langle b^{\{\}}.\mathbf{Rec}\, \overline{b}_{\{\}}.b^{\{\}}.\mathbf{0} \mid c^{\{k\}}.\mathbf{0} \mid \overline{a}_{\{k\}}.a^{\{\}}.\mathbf{0},\ \emptyset\rangle$ we obtain two configurations with different interpretations:

$$HL((M_1 \mid N) \setminus \{b, c\}) = \{\varepsilon, \overline{a}, \overline{a}a\} \qquad HL((M_2 \mid N) \setminus \{b, c\}) = \{\overline{a}, \overline{a}a\}$$

Therefore the parallel composition of CHC configurations cannot be used to define the composition of handshake languages. In order to compose handshake protocols, some more knowledge on the branching structure is needed. CHC provides a suitable formalism to study this structure.

4 Handshake Petri Nets

We argued that not all handshake languages can be represented by CHC configurations, as, for instance, there are non recursive handshake languages. To show the expressive power of our calculus, we provide an alternative semantics of CHC based on Petri nets. In [6], we studied a Petri net representation of handshake protocols, called handshake Petri net, and we showed that all handshake languages can have a, possibly infinite, handshake Petri net representation. In this paper we show that every *finite* handshake Petri net is weakly bisimilar to a CHC configuration.

In this section we introduce handshake Petri nets. The present definition is slightly different from the one in [6], but the results of that paper carry over. We assume some basic knowledge on Petri nets, which we will use in their standard graphical representation [15]. Throughout the paper we will consider Petri nets in their *unsafe* version, where places are allowed to contain several tokens at the same time. This is not just for convenience. Unsafe nets are necessary to carry out our construction.

4.1 Definition

Handshake Petri nets are Petri nets with a special "external interface", reflecting the structure of handshake ports. We define handshake ports in two phases. We first define the static structure of ports, and then we specify the markings.

Let G be a Petri net and let N_O and N_I be a partition of its nodes (places and transitions). The elements of N_O will be called *output transitions / places* while the elements of N_I will be called *input transitions / places*. We give an inductive definition of a *static handshake port* $a = \langle G, N_O, N_I\rangle$ as follows:

– (Basic cases) N_O and N_I contain no transition;
– (Inductive cases) let $a' = \langle G', N'_O, N'_I\rangle$ be a static port:
  - (input prefixing) given a place $p \in N'_I$ with no outgoing arcs, a is obtained from a' by adding an input transition t and an arc from p to t;

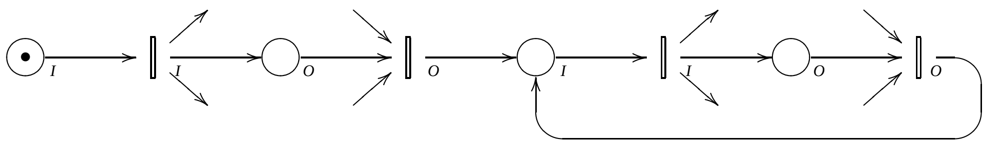


Fig. 3. A passive port (I is for input and O is for output)

- (output prefixing) given a place $p \in N'_O$ with no outgoing arcs and a place $p' \in N'_I$ with exactly one outgoing arc, a is obtained from a' by adding an output transition t, an arc from p to t and an arc from t to p';
- (alternation) given a place $p \in N'_O$ and a transition $t \in N'_I$ with no outgoing arcs, a is obtained from a' by adding an arc from t to p.

Let a' be a static port and let p be a place of (the Petri net of) a' such that if p is an input place, p has an outgoing arc. Let a be the net obtained from a' either by adding one token to p or by keeping a' with no tokens, then a is a *handshake port*. Moreover, if a is as a' (no tokens) or if p is an output place we say that a is an *active port*, otherwise we say that a is a *passive port*.

Let G be a Petri net, G' a subgraph of G and $a = \langle G', N_O, N_I \rangle$ a port s.t. :

- a place of G' may only be connected to transitions of G';
- a transition $t \in N_O$ of G' may only have outgoing arcs to places of G';
- a transition $t \in N_I$ of G' may only have incoming arcs from places of G'.

then a is a *handshake port of G*. Figure 3 shows an example of a passive handshake port of some net. The arrows without source or target indicate the way the port may connect to the rest of the net. The statical structure imposes alternation between the firings of input and output transitions. It is also ensured that, if an input place may ever contain a token, then it must have an outgoing arc to an input transition (receptiveness). Finally, by allowing a port to contain several input and output transitions we are able to model each event separately. For instance, two distinct input transitions may connect differently to the rest of the net, thus providing different resources.

Definition 8. *The pair $H = \langle G_H, Ports_H \rangle$ is a* handshake Petri net (hpn) *just when G_H is a Petri net and $Ports_H$ a set of disjoint handshake ports of G_H.*

Let $H = \langle G_H, Ports_H \rangle$ and let t (p) be a transition (place) of G_H. Then t (p) is *internal* of H if it is neither of input nor of output (in some port a of H).

4.2 Composition

A *linkage* between an active port $a \in Ports_H$ and a passive port $b \in Ports_H$ of an hpn $H = \langle G_H, Ports_H \rangle$ is the hpn $L(H, a, b) = \langle G_{H,a,b}, Ports_H \backslash \{a, b\} \rangle$, where $G_{H,a,b}$ is the net obtained by adding two fresh places p_1 and p_2 to G_H and arcs from each output transition of a to p_1, from p_1 to each input transition of b, from each output transition of b to p_2 and from p_2 to each input transition of a.

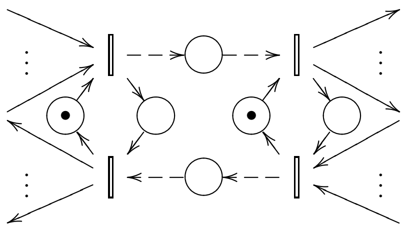


Fig. 4. Example of composition of ports

We call *link* of $L(H, a, b)$, denoted $link(H, a, b)$, the graph consisting of the graphs of a and b plus p_1, p_2 and any arc connecting them to transitions of a or b. Figure 4 shows an example of link between an active port (left) and a passive port (right).

Definition 9. *Let $H_1 = \langle G_1, Ports_1\rangle$ and $H_2 = \langle G_2, Ports_2\rangle$ be two handshake Petri nets. Let $\{a_1, \ldots a_n\} \subseteq Ports_1 \cup Ports_2$ be a set of active handshake ports and let $\{b_1, \ldots b_n\} \subseteq Ports_1 \cup Ports_2$ be a set of passive handshake ports, such that for $1 \leq i \leq n$, $a_i \in Ports_1$ if and only if $b_i \in Ports_2$. Then:*

$$\begin{gathered} H_1 \mid\mid_{\{(a_1,b_1),\ldots(a_n,b_n)\}} H_2 \\ = \\ L(\ldots L(\langle G_1 + G_2, Ports_1 \cup Ports_2\rangle, a_1, b_1), \ldots a_n, b_n) \end{gathered}$$

is the composition of H_1 with H_2 by linking the pairs $(a_1, b_1), \ldots (a_n, b_n)$.

It is easy to see that the composition of two hpns is well-defined and associative.

4.3 Observational Properties of hpns

Let us label a ($\overline{a}$) each input (output) transition of port a, for any $a \in Ports_H$, and τ each internal transition of H. We write $H \xrightarrow{l} H'$ (which reads H l-reduces to H') if a transition labeled l is enabled in G_H and its firing leads to the hpn H'. Seen as labeled transition systems, hpns naturally inherit many definitions that we gave for CHC. Two of these are strong ($\sim$) and weak ($\approx$) bisimilarity. The generality of these definitions allow us to go as far as saying that an hpn is (weakly or strongly) bisimilar to a handshake configuration.

Another inherited definition is that of the function HL, which is identical to the one we gave in Section 3.4 for a handshake configuration. However, we still need to adapt HS. Let $H = \langle G_H, Ports_H\rangle$ be an hpn and let $d_H : Ports_H \rightarrow \{!, ?\}$ be the function which maps each port a to !, when a is active in H, or to ?, when it is passive. We define $HS(H) = \langle Ports_H, d_H\rangle$.

Proposition 10. **[6]** *Let H be a handshake Petri net, then $HL(H)$ is a handshake language on the handshake structure $HS(H)$.*

Proposition 11. **[6]** *Let σ be a handshake language on a handshake structure $\langle P, d\rangle$. Then there is an hpn H_σ such that $HS(H_\sigma) = \langle P, d\rangle$ and $HL(H_\sigma) = \sigma$.*

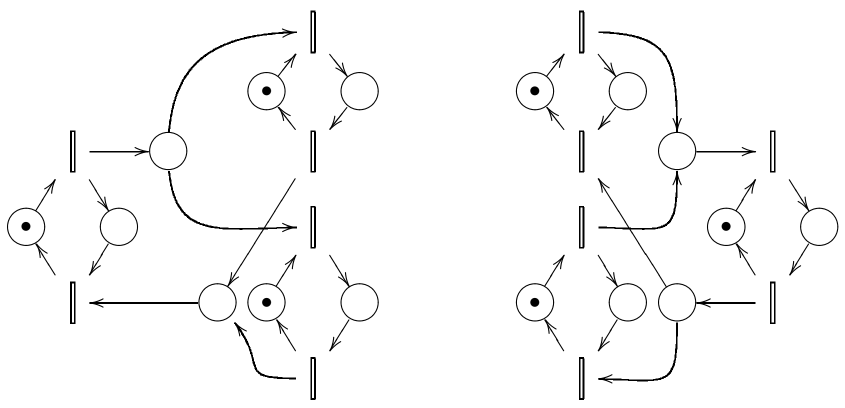


Fig. 5. *OR* (left) and *MIX* (right) handshake components

Note that, for the last result, the net H_σ may contain an infinite number of internal places and transitions. This is unavoidable as languages are in general not recursive and thus not finitely representable. Figure 5 shows two simple examples of handshake protocols: *OR* and *MIX* described in Section 3.3.

5 Full Abstraction and Definability

In this section we relate the calculus CHC with its Petri nets model . We only sketch the constructions, the detailed proofs are available on the extended version [5].

5.1 Full Abstraction

Let M be a handshake configuration, such that $M \rhd \Gamma$. We can assume $M = \langle P, S \rangle$ and define the hpn $[\![M]\!]_\Gamma$ by cases of P. The definitions for open configurations are identical, $[\![\wr P, S \wr]\!]_\Gamma = [\![\langle P, S \rangle]\!]_\Gamma$.

Let $P = \mathbf{0}$. Then $\Gamma = !a$ for some channel a. Now, define a port which contains a single output place p holding a token and call it a as the channel. Let G be the Petri net which contains p plus an internal place q, labeled r, for each $r \in S$, where q contains as many tokens as there are occurrences of r in S. Then $[\![M]\!]_\Gamma$ is the hpn $\langle G, \{a\} \rangle$.

Let $P = a^{\{r_1,\ldots r_n\}}.P'$. Then the last applied typing rule is (inpref), then $\Gamma = ?a$ and $P' \rhd !a$. Let $[\![\langle P', S \rangle]\!]_{!a} = \langle G', Ports' \rangle$. By construction, $Ports' = \{a\}$ where a is an active port. Let p' be the place of a with a token. Let's extend a by adding a fresh input place p, a fresh input transition t labeled a and arcs from p to t and from t to p', then by removing the token from p' and putting a token into p. Finally let's add arcs from t to any place labeled r_i, for $1 \leq i \leq n$. If any of these places does not exist yet, add it anew. We thus obtain a graph G. Then $[\![M]\!]_\Gamma = \langle G, \{a\} \rangle$. The case of the output prefix is dual.

Let $P = \mathbf{Rec}P'$. Then $\Gamma = !a$ and $P' \rhd !a$, for some channel a. Let $[\![\langle P', S\rangle]\!]_{!a} = \langle G', Ports'\rangle$. By construction, $Ports' = \{a\}$ where a is also the active port associated to channel a. Let p be the place of a which holds a token. If p has an incoming arc or if any other place in a has two incoming arcs, $[\![\langle P, S\rangle]\!]_\Gamma = [\![\langle P', S\rangle]\!]_{!a}$. Otherwise a must contain a place p' with no outgoing arcs, by construction. Note that both p and p' must be output places, also by construction. Then replace p and p' by a place q obtained by "joining" them. In particular, q must be the new source of p's outgoing arc and the new target of p''s incoming arc. Call G the graph so obtained. Then $[\![\langle P, S\rangle]\!]_\Gamma = \langle G, \{a\}\rangle$.

Let $P = (P' \mid P'') \setminus \Delta$. We construct $[\![\langle P, S\rangle]\!]_\Gamma$ in three steps. First let $[\![\langle P', S\rangle]\!]^\Delta_{\Gamma_1}$ be obtained from $[\![\langle P', S\rangle]\!]_{\Gamma_1}$ by renaming each port a such that $(a, a) \in \Delta$, as $a^!$ when $\Gamma_1(a) = !$ and as $a^?$ when $\Gamma_1(a) = ?$. Define analogously $[\![\langle P'', S\rangle]\!]^\Delta_{\Gamma_2}$. Then let $\langle G', Ports\rangle = [\![\langle P', S\rangle]\!]^\Delta_{\Gamma_1} \parallel_{\{(a^!, a^?) \mid a \in \Delta\}} [\![\langle P'', S\rangle]\!]^\Delta_{\Gamma_2}$. Then, for any two distinct places p and p' of G' labeled by the same resource r do the following: substitute p and p' by a single place also labeled by r, having all the arcs of both p and p'; note also that by construction, p and p' contained the same number of tokens k, then put k tokens in the new place as well. Let G be the net so obtained. Then $[\![M]\!]_\Gamma = \langle G, Ports\rangle$.

The semantics is well defined and fully abstract with respect to weak bisimilarity:

Lemma 12. *Let M be a configuration such that $M \rhd \Gamma$. Then $[\![M]\!]_\Gamma$ as defined above is a handshake Petri net and $M \approx [\![M]\!]_\Gamma$.*

Theorem 13 (Full Abstraction). *Let M and M' be two configurations such that $M \rhd \Gamma$ and $M' \rhd \Gamma'$. Then $M \approx M' \iff [\![M]\!]_\Gamma \approx [\![M']\!]_{\Gamma'}$.*

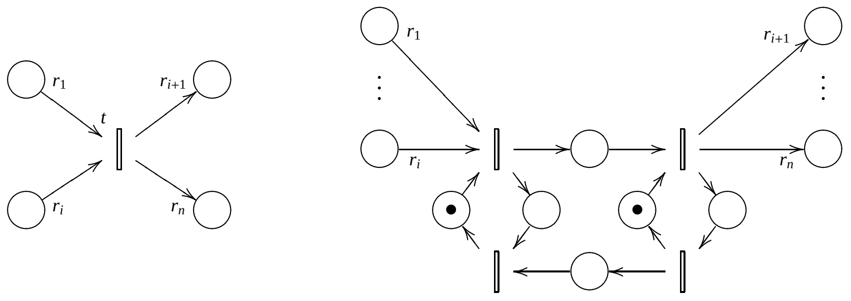
5.2 Definability

For each finite hpn there is a weakly bisimilar handshake configuration:

Theorem 14 (Definability). *Let $H = \langle G, Ports\rangle$ be a handshake Petri net. Then there are a closed handshake configuration M and a handshake type Γ, such that $M \rhd \Gamma$ and $[\![M]\!]_\Gamma \approx H$.*

We present here a simplified construction of the configuration associated to H. The idea is that each port a of the net H can be modeled by a thread $Proc(a, H)$, inductively on the structure of the port.

Each internal transition t is first *unfolded* as a link between two ports and then associated to a process. Let t have incoming arcs from internal places labeled $r_1, \ldots r_i$ and outgoing arcs to internal places labeled $r_{i+1}, \ldots r_n$. Then t is unfolded as follows:



where $r_1, \ldots r_n$ are place labels. . Then let H be a hpn, $u(H)$ is the hpn obtained from H by unfolding each of its internal transitions. It can be shown that $H \approx u(H)$.

For each internal transition t, let l_t be a fresh label associated to it. Consider the following process.

$$Proc(t, H) = (\mathbf{Rec}\ \overline{l}_{t\{r_1,\ldots r_i\}}.l_t.\mathbf{0} \mid l_t^{\{r_{i+1},\ldots r_n\}}.\mathbf{Rec}\ \overline{l_t}.l_t^{\{r_{i+1},\ldots r_n\}}.\mathbf{0}) \setminus \{l_t\}$$

Then we define

$$Proc(H) = Proc(a_1, H) \mid \ldots \mid Proc(a_n, H) \mid Proc(t_1, H) \mid \ldots \mid Proc(t_m, H)$$

where $a_1, \ldots a_n$ are the ports of H and $t_1, \ldots t_m$ are the internal transitions of H. Then let $Conf(H) = \langle Proc(H), S_H \rangle$, where S_H is the multiset of labels of internal places of H with a token and a label appears in S_H as many times as the number of tokens in the corresponding place. Finally let $ch(Ports)$ be the set of names of ports in $Ports$, then $\Gamma_H : ch(Ports) \to \{!, ?\}$ is the function which associates ! to its active ports' names and ? to its passive ports' names. It can be shown that $[\![Conf(H)]\!]_{\Gamma_H} \approx u(H)$. Thus $[\![Conf(H)]\!]_{\Gamma_H} \approx H$.

6 Conclusions

We presented the calculus CHC which describes handshake protocols of communication. We have given it an lts semantics and a Petri nets semantics in terms of handshake Petri nets. We have shown that every finite handshake Petri net corresponds to a closed configuration of the calculus.

We have argued that a branching semantics is necessary to understand handshake protocols, as the trace model cannot properly define composition. The calculus and the two semantics provide the necessary framework to formally study handshake protocols.

Our original aim had been to devise a typing system for CCS, that would ensure the handshake discipline. After many attempts, we came to believe that such typing system would be cumbersome, if at all possible. Consider in particular the MIX component defined in Section 4.3. It is essentially characterized by a form of *inclusively disjunctive* causality: The request sent on the active port causally depends on either of the requests received on the passive ports. However, if both are received, it cannot be established which of the two is actually the cause. This is in contrast with the fact that CCS can be modeled using safe occurrence nets (which correspond to stable event structures), where this kind of causality cannot be represented. Therefore we decided to use a second, different form of communication, in the form of shared resources.

As usual, a result opens new directions to inspect. We would like to characterize the handshake languages that are described by CHC configurations (and thus by finite nets). We would also like to study restrictions on the language or the types to characterize behavioral classes of protocols: deterministic, positional, free choice, etc. It would be interesting to extend the calculus with mobility primitives, like the ones of the π-calculus, and study its expressive power. We also would like to use the calculus to specify and prove the correctness of specific protocols.

Acknowledgments: We thank Roberto Amadio, Catuscia Palamidessi and Frank Valencia for their useful comments. We thank an anonymous referee for spotting a mistake in a previous version of our proofs. The first author would like to thank Simona Ronchi della Rocca, Felice Cardone, Pierre-Louis Curien for their valuable inputs and support. The second author acknowledges the support of the ANR project ParSec: ANR-06-SETI-010-02.

References

1. S. Abramsky, S. Gay, and R. Nagarajan. Interaction categories and the foundations of types concurrent programming. In *Proc. of the 1994 Marktoberdorf Summer School on Deductive Program Design*, pages 35–113. Springer, 1996.
2. S. Arun-Kumar and M. Hennessy. An efficiency preorder for processes. *Acta Informatica*, 29(9):737–760, 1992.
3. A. Bardsley. Balsa: an asynchronous circuit synthesis system. Master's thesis. Department of Computer Science, University of Manchester, 1998.
4. L. Fossati. *Modeling the Handshake Protocol for Asynchrony*. PhD thesis, Dip. di Informatica, Univ. di Torino & Lab. Preuves Programmes et Systèmes (PPS), Univ. Paris 7, 2009.
5. L. Fossati and D. Varacca. The calculus of handshake configurations (extended version). Available at http://www.di.unito.it/ fossati/, 2008.
6. L. Fossati and D. Varacca. A Petri net model of handshake circuits. In *Proc. of First International Workshop on Interactive Concurrency Experience, ICE'08*. ENTCS, Elsevier, 2008. To be published, available at http://www.di.unito.it/ fossati/.
7. D. Gelernter. Generative communication in Linda. *ACM Transactions on Programming Languages and Systems*, 7(1):80–112, 1985.
8. D. R. Ghica. Geometry of synthesis: a structured approach to VLSI design. In *Proc. of POPL '07*, pages 363–375. ACM Press, 2007.
9. http://www.handshakesolutions.com/.
10. M. Josephs, J. Udding, and Y. Yantchev. Handshake algebra. Technical Report SBU-CISM-93-1, School of Computing, Information Systems and Mathematics, South Bank University, London, 1993.
11. N. Kobayashi, B. Pierce, and D. Turner. Linearity and the π-calculus. *ACM Transactions on Programming Languages and Systems*, 21(5):914–947, 1999.
12. I. Mackie. The geometry of interaction machine. In *Proc. of POPL '95*, pages 198–208. ACM Press, 1995.
13. R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989. Second Edition (1991).
14. R. Milner and D. Sangiorgi. Techniques for "weak bisimulation up-to". Revised version of a paper appeared in *Proc. of* CONCUR '92, LNCS 630. Available on Sangiorgi's webpage.
15. W. Reisig. *Petri Nets: An Introduction*, volume 4 of *Monographs in Theoretical Computer Science. An EATCS Series*. Springer, 1985.
16. A. W. Roscoe. Unbounded nondeterminism in CSP. *Journal of Logic and Computation*, 3(2):131–172, April 1993. Previously appeared in 'Two Papers on CSP', tech. monograph PRG-67, Oxford University Computing Laboratory, July 1988.
17. J. Udding. *Classification and Composition of Delay-Insensitive Circuits*. PhD thesis, Department of Math. and C.S., Eindhoven University of Technology, Eindhoven, 1984.
18. K. Van Berkel. *Handshake Circuits: an Asynchronous Architecture for VLSI Design*, volume 5 of *Cambridge International Series on Parallel Computation*. Cambridge University Press, 1993.
19. N. Yoshida, K. Honda, and M. Berger. Linearity and bisimulation. In *Proc. of FoSSaCs '02*, volume 2303 of *LNCS*, pages 417–433. Springer, 2002.