A Petri Net Model of Handshake Protocols

Luca Fossati[1]

Dipartimento di Informatica - Univ. di Torino, Italia
PPS - CNRS & Univ. Paris Diderot, France

Daniele Varacca

PPS - CNRS & Univ. Paris Diderot, France

Abstract

We propose a Petri net model of handshake protocols. These are asynchronous communication protocols which enforce several properties such as absence of transmission interference and insensitivity from delays of propagation on wires. We introduce the notion of *handshake Petri net*, a Petri net with a specific external interface. We show that the set of observable *quiescent* traces generated by such a net captures the properties defining a handshake protocol. Conversely we show that for any handshake protocol we can construct a corresponding net. We also study different subclasses of the model. Many examples are provided.

Keywords: Handshake protocol, Petri nets, asynchronous communications, delay-insensitivity, transmission interference.

1 Introduction

The asynchronous style of computation is characterized by several subunits acting locally, independently of each other, as opposed to the synchronous style, where a central clock disciplines everything. Working with asynchronous systems, there are a few situations one would like to avoid. One is *transmission interference* which may occur when two consecutive messages are sent over the same channel, with the risk of clashing into one another [14]. Another one is *computation interference*, where a message is delivered to an unready receiver [9,16].

One way to rule out such situations is by adopting communication protocols to enforce the desired behavior. For instance, *delay-insensitive* protocols guarantee that a system's behavior is independent of propagation delays over wires and of computational speeds of single units, thus preventing computation interference. Among those, we focus on the *handshake protocol* which requires that each message sent is followed by an acknowledge, thus preventing transmission interference.

[1] Corresponding author: fossati@di.unito.it

Thanks to its simplicity and efficiency, the handshake protocol has gained the interest of enterprises like Sun and Philips [6]. However, little research has been put forward on foundational aspects. For quite a few years the foundational research on handshake circuits, circuits obeying the handshake protocol, has relied on the model introduced by Kees Van Berkel in his PhD thesis [15]. While Van Berkel's model will continue to be a reference for many aspects, it contains a serious shortcoming: the process composition it defines is not associative, as proved by the first author [3].

To solve this problem the first author [3] proposed a game semantics for handshake circuits which describes their composition correctly for the first time. Technically, the result was accomplished by representing handshake "behaviors" as sums of *deterministic handshake strategies*. The price to pay is that there are behaviors which do not fit in this representation. The crucial example is the mixer component, MIX, which will be described in Section 3.3.

This led us to look for other kinds of models. A graphical representation is probably the most natural choice for dealing with asynchronous circuits: in graphs as in circuits, composition is easy when everything else works properly. Several works have taken a similar perspective ([1],[8],...). In particular Dan Ghica developed a language for asynchronous hardware design by taking inspiration from the Geometry of Interaction and handshake circuits [5]. However his goal was to improve previous hardware design languages [15,2] and not to capture all handshake behaviors.

The model we present in this paper is based on Petri nets [11]. Petri nets are widely used as models of asynchrony, and are close to the context in which the handshake communication protocol originated [12]. However, the properties of delay-insensitivity and absence of transmission interference had not yet been formalized under a graphical representation. We call our model *handshake Petri nets*. We show that handshake Petri nets capture precisely the handshake protocol, in the sense that the behavior of every net is a handshake language and that every handshake language is the behavior of some net.

Plan of the paper

In Section 2 we define the notion of handshake language as set of traces (taking inspiration from [15] and [3]). In Section 3 we introduce handshake Petri nets and some of their subclasses. We put a special emphasis on deterministic behaviors, as well as on those nondeterministic behaviors which cannot be expressed as sums of deterministic components. Finally, in Section 4 we provide an interpretation of handshake Petri nets into handshake languages and we prove the correctness and completeness of this interpretation. Completeness of deterministic handshake Petri nets with respect to deterministic handshake languages will follow as a corollary.

2 The Handshake Protocol

In this section we characterize the handshake protocol in terms of languages obeying its communication discipline. We do not exactly give another trace model as, for instance, we do not define composition. We just need a yardstick against which to measure the correctness of our model. Moreover, we are only interested in the communication discipline, so we assume circuits have *nonput ports* (no data is exchanged in a communication). We leave the more general case for further work.

Definition 2.1 A *handshake structure* is a pair $\langle P, d\rangle$, where P is a finite set of *ports* and the function $d : P \to \{act, pas\}$ 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 $\langle P, d\rangle$ be a handshake structure and let $\cup_{p\in P}\{p, \bar{p}\}$ be the alphabet of messages on $\langle P, d\rangle$. In particular, p and $\bar{p}$ are both messages on some port p[2]. Two messages are *independent* when they are not on the same port. The function λ_P is defined on $\cup_{p\in P}\{p, \bar{p}\}$ so that $\lambda_P(p) = -$ (input message) and $\lambda_P(\bar{p}) = +$ (output message), for all $p \in P$. We may write λ instead of λ_P when P is redundant or clear from the context.

Let t be a trace on the alphabet of messages $\cup_{p\in P}\{p, \bar{p}\}$. t is a *handshake trace* on $\langle P, d\rangle$ if for all $p \in P$:

- $t \restriction \{p, \bar{p}\} = \bar{p}p\bar{p}p\ldots$ when $d(p) = act$;
- $t \restriction \{p, \bar{p}\} = p\bar{p}p\bar{p}\ldots$ when $d(p) = pas$;

We call *thread* each such restriction and we call *request* (*acknowledge*) the message appearing in the odd (even) positions in each thread of p.

Threads induce an equivalence on traces, the *homotopy* relation $\sim_P$. Given two handshake traces s and t, we say that $s \sim_P t$ when they have the same set of threads. As usual, we denote by $[s]_\sim$ the equivalence class of trace s with respect to $\sim$, we call $[s]_\sim$ the *position* of s.

Given a set of traces σ we write $\sigma^\leq$ 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 message σ can output after s:

$$\forall s \cdot m \in \sigma^\leq, \lambda(m) = -.$$

We write $Pas(\sigma)$ for the set of passive traces in $\sigma^\leq$.

We define $\mathbf{r}_P$ as the smallest binary relation which is closed by reflexivity, transitivity and concatenation, and such that for any distinct ports $p, q \in P$:

(i) $p\bar{q}\ \mathbf{r}_P\ \bar{q}p$;

(ii) $\bar{p}\bar{q}\ \mathbf{r}_P\ \bar{q}\bar{p}$;

(iii) $pq\ \mathbf{r}_P\ qp$

We say that s *reorders* t in P if $s\ \mathbf{r}_P\ t$. Note that the relation $\mathbf{r}_P$ is *not* symmetric.

Let s be a handshake trace and $p \in P$. We write $p\ \mathbf{x}_P\ s$ if sp is still a handshake trace. We are now ready for the definition of handshake language.

Definition 2.2 A *(handshake) language* σ on $\langle P, d\rangle$ is a non-empty set of finite handshake traces on $\langle P, d\rangle$ such that:

(i) $Pas(\sigma) \subseteq \sigma$ (closed under passive prefixes);

(ii) $(t \in \sigma \wedge s\ \mathbf{r}_P\ t) \Rightarrow s \in \sigma$ (reorder closed);

[2] One may object that the same name p is used for both the message and the port. However the context will always make clear which p we are referring to.

(iii) $(s \in \sigma^{\leq} \wedge p \,\mathbf{x}_P\, s) \Rightarrow s \cdot p \in \sigma^{\leq}$ (receptive).

Note that the traces of a handshake language are finite, but the language itself may contain an infinite number of traces.

Definition 2.3 Let σ be a handshake language. We say that σ is *positional* if, for finite $s, s' \in \sigma^{\leq}$, with $s \sim_P s'$, we have:

(i) $s \cdot t \in \sigma^{\leq} \Rightarrow s' \cdot t \in \sigma^{\leq}$;

(ii) $s \in \sigma \Rightarrow s' \in \sigma$;

We say that σ is *deterministic* if for any distinct $p, q \in P$:

(i) $s \cdot \bar{p} \in \sigma^{\leq} \Rightarrow s \notin \sigma$ (progress);

(ii) $s \cdot \bar{p} \in \sigma^{\leq} \wedge s \cdot \bar{q} \in \sigma^{\leq} \Rightarrow s \cdot \bar{p} \cdot \bar{q} \in \sigma^{\leq}$ (absence of conflict).

Positionality means that the only thing relevant in a choice is the position we are at and not the way we reached it. As for determinism: when a deterministic language σ is able to produce an output, waiting is not an option; when there is a choice of two outputs, one choice must not exclude the other. It is not difficult to prove the following fact.

Proposition 2.4 *A deterministic language is positional.*

Examples

Consider the handshake structures $\mathcal{P} = \langle \{p\}, \{p \mapsto pas\} \rangle$ and $\mathcal{A} = \langle \{p\}, \{p \mapsto act\} \rangle$, corresponding respectively to a passive and to an active port. Then, $p\bar{p}p\bar{p}p\bar{p}$ is a handshake trace on $\mathcal{P}$ but not on $\mathcal{A}$. The set

$$\{p\bar{p}, p\bar{p}p\bar{p}, p\bar{p}p\bar{p}p\bar{p}, \ldots\}$$

is not closed under passive prefixes as it does not contain the empty string, then it is not a handshake language. Whereas both sets

$$RUN_p = \{\bar{p}, \bar{p}p\bar{p}, \bar{p}p\bar{p}p\bar{p}, \ldots\} \qquad \text{and} \qquad \{\varepsilon, \bar{p}, \bar{p}p\bar{p}, \bar{p}p\bar{p}p\bar{p}, \ldots\}$$

are handshake languages on $\mathcal{A}$. In particular RUN_p is deterministic, the other is not. The set

$$\{\bar{p}, \bar{p}p\bar{p}, \bar{p}p\bar{p}p\bar{p}\}$$

is not a handshake language on $\mathcal{A}$, because it is not receptive: after the last trace the environment is still supposed to send an acknowledge, but the language is not ready to receive it. Even the receptive RUN_p becomes not receptive if we extend its structure with a passive port, as in $\mathcal{B} = \langle \{p, q\}, \{p \mapsto act, q \mapsto pas\} \rangle$. A process which is receptive with respect to $\mathcal{B}$ is the following:

$$REP_{p,q} = \{\varepsilon, q\bar{p}, q\bar{p}p\bar{p}, q\bar{p}p\bar{p}p\bar{p}, \ldots\}$$

this process is also called repeater since, after reception of a request on its passive port it "handshakes" indefinitely on the active. Now look at the following sets on $\mathcal{B}$:

$$\{\varepsilon, q\bar{q}\bar{p}, q\bar{q}\bar{p}q, q\bar{q}\bar{p}qp\}$$

$$\{\varepsilon, q\bar{q}\bar{p}, q\bar{p}\bar{q}, q\bar{q}\bar{p}q, q\bar{p}\bar{q}q, q\bar{q}\bar{p}p, q\bar{p}\bar{q}p, q\bar{q}\bar{p}qp, q\bar{q}\bar{p}pq, q\bar{p}\bar{q}pq, q\bar{p}\bar{q}qp\}$$

Neither of them is reorder-closed, then neither of them is a handshake language. For example, $q\bar{q}q\bar{p}\ \mathbf{r}_{\{p,q\}}\ q\bar{q}\bar{p}q$ but $q\bar{q}\bar{p}q$ is in the prefix-closure of both of the above sets, while $q\bar{q}q\bar{p}$ is in the prefix-closure of none. We leave it to the reader to figure out the reorder-closures of the above two sets and to show that the second's is a handshake language while the first's is not.

Finally, consider yet another set on $\mathcal{B}$:

$$\{\varepsilon, q\bar{p}, q\bar{q}, q\bar{p}p\bar{q}, q\bar{q}q\bar{p}, q\bar{p}p\bar{q}q, q\bar{q}q\bar{p}p\bar{p}, q\bar{q}q\bar{p}p\bar{p}p\}$$

The reader can verify that it satisfies all the properties of a handshake language, we show that it does not satisfy those of positionality. Note that $q\bar{p}p\bar{q}q$ and $q\bar{q}q\bar{p}p$ are two traces with the same position and that both are in the prefix-closure of the above set. However, while the first is actually an element of the set, the second is not and, conversely, while the second can be extended with $\bar{p}$, the first cannot. The language is not deterministic either since after the initial q there is a mutually exclusive choice between $\bar{p}$ and $\bar{q}$.

3 Handshake Petri Nets

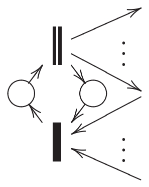
We assume some basic knowledge on Petri Nets, which we will use in their standard graphical representation [11]. Throughout the paper we will consider Petri nets in their *unsafe* version, where places are allowed to contain several tokens at the same time. This is not just for convenience. Unsafe nets are necessary to carry out our construction. We also stress that the nets we consider are in general not finite, in the sense that they may have infinitely many places and/or transitions.

Handshake Petri nets are characterized by a special "external interface" which reflects the structure of handshake ports. Let I and O be disjoint finite subsets of the set of transitions of a Petri net G. We call the triple $\langle G, I, O\rangle$ an *interacting Petri net (ipn)*, where the transitions in I are its *input transitions* and those in O are its *output transitions*. A transition is *external* if it is an input or an output transition, *internal* otherwise.

An ipn $\langle G, I, O\rangle$ *t-reduces* to $\langle G', I, O\rangle$, $\langle G, I, O\rangle \xrightarrow{t} \langle G', I, O\rangle$, when the transition t can fire in G and the result of the firing is G'. We call *execution* of $\langle G, I, O\rangle$ any sequence of transition firings starting from $\langle G, I, O\rangle$.

Definition 3.1 An ipn $\langle G, I, O\rangle$ is a *handshake Petri net (hpn)* when:

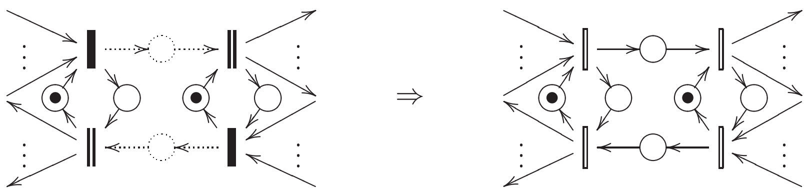
- I and O have the same cardinality;
- input (output) transitions have exactly one incoming (outgoing) arc;
- each input transition is paired with an output transition by means of the following structure:



where the input transition is denoted by two parallel bars and the output transition by one thick bar. Such a structure represents a *port*;

- at any time, each port contains exactly one token. In particular, when the token enables the input transition the port is *passive*, when it enables the output transition it is *active*.

Two hpns may be composed by linking a set of ports of the first net with a set of ports of the second net. Each "link" must be between a passive and an active port and is done by adding two new places (and four new arcs) between them, as follows:



On the left we represent the two ports before the link is done, where the new arcs and places are not there yet: we draw them dashed to indicate this. On the right we represent the situation after the link, note that we use a different graphical notation for the ports' transitions as, after composition, they become internal. In fact, the new net will have as external transitions all and only those external transitions of the two original nets that have not been linked during the composition, and each of these inherited external transition will keep its status: inputs will stay inputs and outputs will stay outputs. It is easy to see that the composition of two hpns is still an hpn, moreover composition of hpns is trivially associative.

In the rest of this section we will show several examples of standard handshake components represented as handshake Petri nets. We will present each example within a specific subclass of the general model.

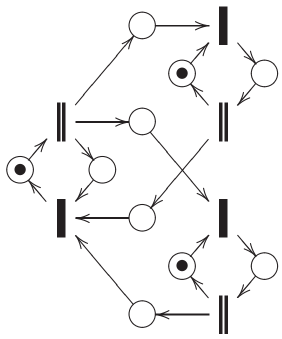
3.1 Handshake Marked Graphs

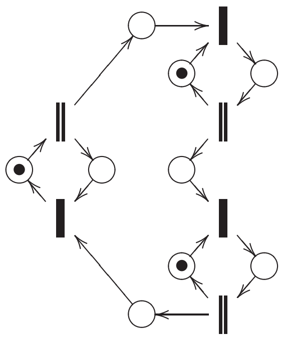
In the first stage we focus on marked graphs [10], which are Petri nets where each place has at most one incoming and at most one outgoing arc.

Marked graphs are significant as they allow to identify places with communication channels and in turn to represent all and only circuits which can be built out of channel, synchronization and fork operations. Moreover they have a special historical importance for the handshake protocol [13]. We call *handshake marked graph* a handshake Petri net which is also a marked graph.

Examples

Marked graphs represent the core of determinism. In particular they allow the representation of most deterministic handshake components: *STOP*, *RUN*, *CON*, *SEQ*, *PAR*, *PAS*, *JOIN* (in the notation of [15]). Two of these components are represented below[3].





PAR (left) waits for a request on its passive port and then starts two handshakes in parallel on its active ports. Only after successful termination of both it acknowledges to the first request. *SEQ* (right) also waits for a request on its passive component, but then it starts its active ports in sequence, before finally acknowledging to the initial request.

The examples show that handshake marked graphs (or marked graphs in general) always react in the same way to a given stimulus. For example, *SEQ* always sends a request on its first active port after the reception of a request on its passive port. It can be shown that handshake marked graphs embed a particular subclass of handshake languages where each pair stimulus/response can be seen as a couple of brackets in the language and each trace becomes well-bracketed with respect to any of these couples, after a fixed number of closing brackets.

3.2 Deterministic Extensions

Marked graphs express only deterministic behaviors but not all deterministic behaviors are captured by marked graphs. As far as we know, no structural characterization of determinism in Petri nets exists in the literature. We propose a definition that completely characterizes determinism in the context of handshake nets.

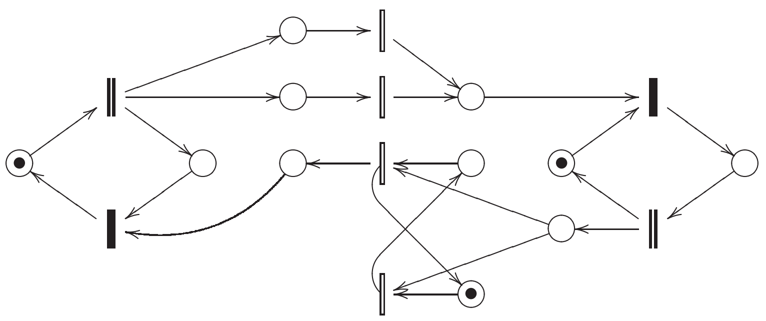
Definition 3.2 A *handshake deterministic-branching net* (a *handshake DB net*, or just a *DB net*, for short) is a handshake Petri net in which every place p with several outgoing arcs is such that:

- Each post-transition t of p has a "guard", a place whose only post-transition is t;
- Exactly one of the guards of p's post-transitions initially contains a token, so that at most one post-transition may initially be enabled;
- Each of p's post-transitions has exactly one outgoing arc to some guard of a post-transition of p, so that each time one post-transition has fired one post-transition may be enabled;
- Each guard of a post-transition of p may have incoming arcs only from p's post-transitions, so that no more than one post-transition may ever be enabled.

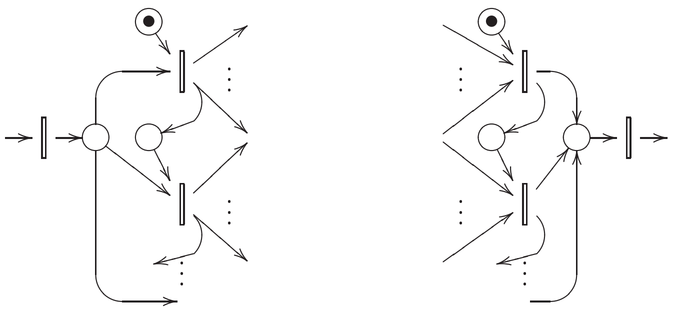
[3] Although handshake Petri nets are formalized here for the first time, a similar representation for both components had already been given as far back as [12]. Actually, those pictures were an inspiration for this work.

Examples

As an example, consider $COUNT_N$ which, after reception of a request on its passive port, handshakes N times on its active port. Then it acknowledges to the first request and returns to wait for an activation. In this case, the circuit needs to decide (deterministically, of course) when to acknowledge to a passive request (after N handshakes on the active port). Here is the circuit $COUNT_2$, also known as DUP:



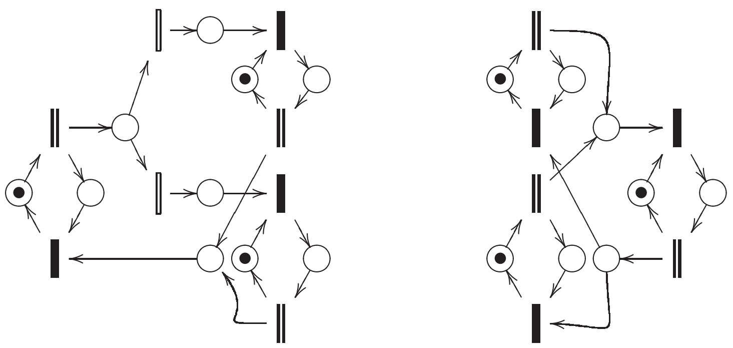
A DB structure allows us to select each firing of a given transition and associate it to a brand new dedicated transition, as shown below:



We call the above *input* (left) and *output* (right) *occurrence selecters.*

3.3 General nets

In this subsection we present two examples of nondeterministic nets, the OR (below left) and the MIX (below right) handshake components:



OR has a passive and two active ports. When a request on the passive port arrives a request on either active port is sent. Acknowledge to this last request enables an acknowledge to the first one. As the picture shows, this example can be modeled by a *free choice* net (transitions with a shared precondition do not have any preconditions other than that).

Conversely, MIX has two passive and one active port. Each time an environment request arrives (on either passive port) MIX handshakes on its active port and after completion it acknowledges to the first request. If by the time the handshake on the active port completes the environment had sent a request on the other port, MIX chooses nondeterministically which request to acknowledge first.

This situation could not be described with a free-choice net since the choice of which request to acknowledge may not be a choice at all if the environment only sent one request.

4 Soundness and Completeness

Consider an hpn $H = \langle G, I, O \rangle$, name its ports and let P_H be the set of these names. Now take $d_H : P_H \to \{act, pas\}$, the function which maps each port name p to the appropriate label, *act* if port p is active in H, *pas* if it is passive. This allows us to define the handshake structure $HS(H) = \langle P_H, d_H \rangle$.

Then for any port p, name p ($\bar{p}$) its input (output) transition, and name τ any internal transition. An execution t of H is *quiescent* when for no $p \in P_H$ it can be extended as $H \xrightarrow{t}\!\!\!\!\rightarrow \xrightarrow{(\tau)^*}\!\!\!\!\rightarrow \xrightarrow{\bar{p}}$. We define $HL(H)$ as the set of strings consisting of the external restriction of each quiescent execution of H.

The main results of this paper, the soundness and completeness of the Petri nets model, can be respectively formalized by the following theorems.

Theorem 4.1 *Let H be a handshake Petri net, then $HL(H)$ is a handshake language on the handshake structure $HS(H)$.*

Theorem 4.2 *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 $HL(H_\sigma) = \sigma$.*

The proof of soundness is a rather straightforward verification of the properties defining a handshake language. In the remaining of this section, we try to hint the proof of completeness (theorem 4.2). We must warn that the construction of H_σ we propose may lead to an infinite net, but let us also point out that an infinite representation is in general unavoidable. An example of language with no finite representation is the one which contains an infinite chain of (finite) traces where outputs are chosen according to a non recursive function.

Let us focus first on positional handshake languages. These languages make their choices according to the reached position, regardless of the particular interleaving followed in the execution. In the following we write p_i ($\bar{q}_i$) for the ith occurrence of input p (output $\bar{q}$). Then we can represent a *choice* as a pair made of a position $[s]_\sim$ and an output occurrence or a special symbol $*$, where $\langle [s]_\sim, \bar{q}_i \rangle$ expresses the choice of "playing" $\bar{q}_i$ at $[s]_\sim$ and $\langle [s]_\sim, * \rangle$ the choice of doing nothing at $[s]_\sim$.

Let σ be a handshake language and c a choice in σ. Let $t \in \sigma^{\leq}$, we say that t *allows* c in σ ($t \rightarrow_\sigma c$) when $[t]_\sim = fst(c)$ and:

- $t \in \sigma$ if $snd(c) = *$;
- $t \cdot snd(c) \in \sigma^{\leq}$ otherwise.

We say that t *prevents* c in σ ($t \nrightarrow_\sigma c$) when it does not allow it.

If we consider positional strategies we see immediately that positions, rather than traces, allow choices. Moreover, since we only consider data-less communications and since outputs do not affect choices (by reordering), a position can be represented by a set of occurrences of distinct input messages, taking the last input occurrence of each thread.

Starting from the above observations and systematically using the selecter structures introduced in Section 3.2 to select occurrences of input and output messages we are able to construct a handshake Petri net which corresponds to the given positional handshake language.

Proposition 4.3 *Let σ be a positional handshake language on $\langle P, d\rangle$. Then there is an hpn H_σ such that $HS(H_\sigma) = \langle P, d\rangle$ and $HL(H_\sigma) = \sigma$.*

Since the construction associates single occurrences to transitions and since a move may occur infinitely many times, the constructed graph H_σ is in general infinite.

In the non-positional case, reshuffling the threads of a trace may affect a choice. We first define an *atomic reshuffling* of a trace $t(= t' \cdot m \cdot n \cdot t'')$ as a trace of the form $t' \cdot n \cdot m \cdot t''$, for m and n independent messages.

Definition 4.4 Let S be a set of pairs of the form $\langle p_i, \bar{q}_j\rangle$. S is *critical* for a choice c in a handshake language σ just when, for all $t \in \sigma^{\leq}$ such that $[t]_\sim = fst(c)$,

$$\forall \langle p_i, \bar{q}_j\rangle \in S, t = t' \cdot \bar{q}_j \cdot t'' \cdot p_i \cdot t''' \;\Rightarrow\; t \nrightarrow_\sigma c.$$

We write $crit_\sigma(c)$ for the set of minimal critical sets for c in σ.

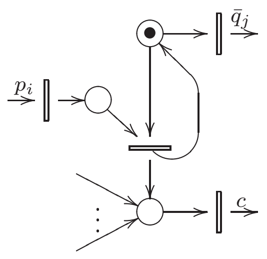
A similar notion is that of *critical pair* (the above being critical set of pairs) for c in σ: a pair $\langle p_i, \bar{q}_j\rangle$ for which there is $s \in \sigma$ such that $s = s' \cdot \bar{q}_j \cdot p_i \cdot s'' \nrightarrow_\sigma c$ while $s' \cdot p_i \cdot \bar{q}_j \cdot s'' \rightarrow_\sigma c$. If $\langle p_i, \bar{q}_j\rangle$ is a critical pair for c in σ, we say that it is *inverted* in t if and only if $t = t' \cdot \bar{q}_j \cdot t'' \cdot p_i \cdot t'''$.

Lemma 4.5 *Let c be a choice in a handshake language σ. Let $t \in \sigma^{\leq}$, $[t]_\sim = fst(c)$:*

$$t \nrightarrow_\sigma c \iff \exists S \in crit_\sigma(c), \forall \langle p_i, \bar{q}_j\rangle \in S, t = t' \cdot \bar{q}_j \cdot t'' \cdot p_i \cdot t'''$$

Proof (Sketch) The direction right-to-left is almost immediate. For the other direction, S is made of all the critical pairs which are inverted in t. Then we can take any trace s with the same threads as t and in which all the pairs of S are inverted and prove that $s \nrightarrow_\sigma c$ by induction on the number of atomic reshuffling needed to change s into t. Note in particular that if an atomic reshuffling affects a choice, then it must consist of a critical pair which is inverted, as all the others are reorderings. So S is critical, if it is not minimal we can take the minimal critical set contained in S and we are done. □

The construction of the net is a modification of the one for positional handshake languages, as we briefly sketch here. For any minimal critical set S for c in σ and for all $\langle p_i, \bar{q}_j \rangle \in S$ we connect transitions p_i, $\bar{q}_j$ and c as follows:



Suppose that the jth firing of $\bar{q}$ occurs before the ith firing of p, and similarly for all the other pairs in S (represented in the picture by the other incoming arcs in the precondition of transition c). Then c is clearly prevented. Note that this scheme works for both a choice to output and a choice not to, as each choice corresponds to a transition in the graph.

The completeness theorem, specializes to several classes of nets and languages. For instance to the deterministic case.

Theorem 4.6 *Let H be a handshake DB net, then $HL(H)$ is a deterministic handshake language on the handshake structure $HS(H)$. Conversely, if σ is a deterministic handshake language on $\langle P, d \rangle$, there is a DB net H_σ such that $HS(H_\sigma) = \langle P, d \rangle$ and $HL(H_\sigma) = \sigma$.*

As we mentioned, marked graphs correspond to a particular class of languages too, the *well-bracketed* ones. In this case there is even a construction yielding finite graphs. We still do not know any independent characterization of the nets that correspond to positional languages.

5 Conclusions

In this paper we presented a version of Petri nets, featuring a particular structure on external connections, that models the handshake protocol of communication. We showed that this model embeds the model of handshake games and strategies [3], but is more expressive.

Graphical representations as Petri nets are very close to the reality of circuits and are useful in explaining the circuits' dynamics. Compared to trace models like games and strategies they go one level deeper: channels are unidirectional (as it usually happens in circuits) and bidirectionality can be obtained by pairing.

This higher level of intensionality brings us to reconsider the model of handshake Petri nets, which can be seen not only as a semantic model for handshake circuits, but also as their syntax. Carrying on this track we could attempt to provide a normal form for handshake Petri nets since, as we have seen, two different nets may have the same behavior.

Also, the graphical representation may drive the definition of a more standard notion of syntax in the form of a process calculus. Van Berkel already proposed the *handshake process calculus* [15] (see also [7]). However their goal was not to capture all possible handshake behaviors and a complete process calculus of handshake circuits is still wanted.

In a recent proposal [4] we define what we call the *calculus of handshake configurations* and we show that it is complete with respect to handshake languages as defined in Definition 2.2.

Further directions include a deeper analysis of data-exchange, as in this paper we focused especially on data-less communications. It would also be interesting to exploit those subclasses which allow finite representations of a given subset of handshake languages, as we tried to hint in Section 3.1.

Note finally that in recent times the foundational research on the field is starting to awake again: besides [3], other efforts have been made to apply game semantics to the synthesis of HDLs for handshake circuits [5]. This strengthens our belief on the importance of the communication protocol inside the universe of asynchrony, the idea that the current model is just the core of a larger representation.

Acknowledgments

The first author would like to thank Simona Ronchi della Rocca, Felice Cardone, Pierre-Louis Curien for interesting discussions and for their support. Thanks to Gianfranco Balbo, Marco Beccuti, Gianfranco Ciardo and Lucia Pomello for their advice on Petri nets.

References

[1] S. Abramsky, S. Gay, and R. Nagarajan. Interaction categories and the foundations of types concurrent programming. In M. Broy, editor, *Proceedings of the 1994 Marktoberdorf Summer School on Deductive Program Design*, pages 35–113. Springer, 1996.

[2] A. Bardsley. Balsa: an asynchronous circuit synthesis system. Master's thesis, Department of Computer Science, University of Manchester, 1998.

[3] L. Fossati. Handshake games. In I. Mackie, editor, *Proceedings of DCM'06*, volume 171-3, pages 21–41. ENTCS, Elsevier, 2007.

[4] L. Fossati and D. Varacca. A calculus for handshake configurations. submitted to FoSSaCS'09.

[5] Dan R. Ghica. Geometry of synthesis: a structured approach to VLSI design. In *Proceedings of POPL'07*, pages 363–375. ACM Press, 2007.

[6] http://www.handshakesolutions.com/.

[7] M.B. Josephs, J.T. Udding, and Y. Yantchev. Handshake algebra. Technical Report SBU-CISM-93-1, School of Computing, Information Systems and Mathematics, South Bank University, London, 1993.

[8] Ian Mackie. The geometry of interaction machine. In *Proceedings of POPL'95*, pages 198–208. ACM Press, 1995.

[9] C.E. Molnar, T.-P. Fang, and F.U. Rosenberg. Synthesis of delay-insensitive modules. In *Proceedings of the 1985 Chapel Hill Conference on Very Large Scale Integration*, pages 67–86. Computer Science Press, 1985.

[10] James Lyle Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice Hall PTR, 1981.

[11] Wolfgang Reisig. *Petri Nets: An Introduction*, volume 4 of *Monographs in Theoretical Computer Science. An EATCS Series*. Springer, 1985.

[12] The Computation Structures Group. Progress report 1969-70. Technical Report MIT-MAC-memo-53, Massachussets Institute of Technology, Project MAC, 1972.

[13] The Computation Structures Group. Progress report 1970-71. Technical Report MIT-MAC-memo-64, Massachussets Institute of Technology, Project MAC, 1972.

[14] J.T. Udding. *Classification and Composition of Delay-Insensitive Circuits.* PhD thesis, Department of Math. and C.S., Eindhoven University of Technology, 1984.

[15] K. Van Berkel. *Handshake Circuits: an Asynchronous Architecture for VLSI Design*, volume 5 of *Cambridge International Series on Parallel Computation.* Cambridge University Press, 1993.

[16] T. Verhoeff. Characterizations of delay-insensitive communication protocols. Computing Science Notes '89/6, Department of Math. and C.S., Eindhoven University of Technology, Eindhoven, May 1989.

A Appendix

We state two definitions which will be useful in the proofs which follow.

Definition A.1 Let s and t be two handshake traces on a given handshake structure, such that $s \sim t$. We define $d_\sim(s,t)$, the *homotopy distance* between s and t as follows:

- $d_\sim(s,t) = 0 \iff s = t$
- $d_\sim(s,t) = 1 \iff (s = s' \cdot m \cdot n \cdot s'') \wedge (t = s' \cdot n \cdot m \cdot s'')$, for two messages m and n;
- In general $d_\sim(s,t) = k > 0$ if and only if $s \neq t$ and there is a sequence of $k+1$ (and no less) traces $s_0, s_1, \ldots s_k$ such that $s = s_0$, $s_k = t$ and for all $0 \leq i < k$, $d_\sim(s_i, s_{i+1}) = 1$.

The second definition is less significant, but still useful. Let H be an hpn and t and t' two distinct external transitions of H. We call t' the *complement* of t, and viceversa, when t and t' belong to the same port.

Proof of Proposition 2.4 Let σ be a deterministic handshake language and let $s, t \in \sigma^{\leq}$, such that $s \sim t$. Suppose that $s \cdot \bar{s} \in \sigma^{\leq}$. We show that $t \cdot \bar{s} \in \sigma^{\leq}$ by induction on $d = d_\sim(s,t)$:

- $d = 0$. $s = t$, then $t \cdot \bar{s} \in \sigma^{\leq}$;
- $d = l + 1$. There is a sequence $s = t_0, \ldots t_{l+1} = t$ such that $d_\sim(t_i, t_{i+1}) = 1$ and $t_i \cdot \bar{s} \in \sigma^{\leq}$, $\forall 0 \leq i \leq l$. Let $t_l = t' \cdot m \cdot n \cdot t''$ and $t_{l+1} = t' \cdot n \cdot m \cdot t''$. If m is an output or n an input, $t_{l+1} \cdot \bar{s} \mathbf{\ r\ } t_l \cdot \bar{s}$, which implies $t_{l+1} \cdot \bar{s} \in \sigma^{\leq}$. Let m be an input and n an output, then we prove $t_{l+1} \cdot \bar{s} \in \sigma^{\leq}$ by induction on the length of $\bar{s}$.
  · $\bar{s} = \varepsilon$. Then $t_{l+1} \cdot \varepsilon = t_{l+1} \in \sigma^{\leq}$ by hypothesis;
  · $\bar{s} = \bar{s}'a$. $t_{l+1} \cdot \bar{s}' \in \sigma^{\leq}$ by hypothesis. If a is an input, $t_{l+1} \cdot \bar{s}' \cdot a \in \sigma^{\leq}$ by receptivity. Then let a be an output and let $a_1, \ldots a_k$ be all the outputs that σ can send at $t_{l+1} \cdot \bar{s}'$. Then determinism implies $t_{l+1} \cdot \bar{s}' \cdot a_1 \ldots a_k \in \sigma^{\leq}$. Note also that if an output was possible at $t_{l+1} \cdot \bar{s}' \cdot a_1 \ldots a_k$, it would also be possible at $t_{l+1} \cdot \bar{s}'$, by reordering. Then $t_{l+1} \cdot \bar{s}' \cdot a_1 \ldots a_k$ is passive in σ. Then also $t_l \cdot \bar{s}' \cdot a_1 \ldots a_k \in \sigma$, as it reorders $t_{l+1} \cdot \bar{s}' \cdot a_1 \ldots a_k$. Then $a \in \{a_1, \ldots a_k\}$, as a is an output and $t_l \cdot \bar{s}' \cdot a \in \sigma^{\leq}$. Then $t_{l+1} \cdot \bar{s}' \cdot a \in \sigma^{\leq}$.

The proof that $s \in \sigma \Rightarrow t \in \sigma$ is even simpler, as the outer induction alone will do.□

Proof of Theorem 4.1 We proceed by successive steps:

- *$HL(H)$ is a set of finite handshake traces on $HS(H)$.* The structure of handshake ports implies that all observable threads alternate inputs and outputs, starting with an input on passive ports and with an output on active ports. Moreover, executions are finite sequences of firings, then their external restrictions are also finite.
- *$HL(H)$ is non-empty.* By definition an execution is a sequence of firings, then the empty sequence is also an execution and its external restriction is an external

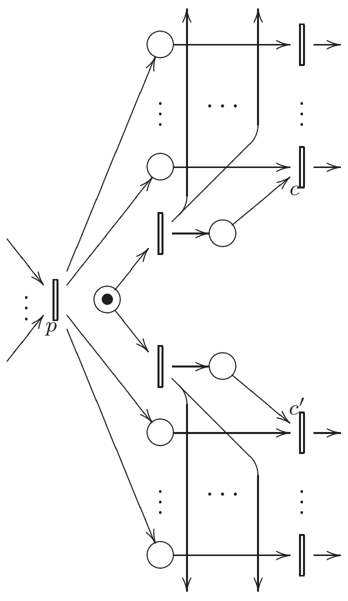
trace. If the empty sequence is not quiescent in H it is the prefix of a quiescent execution (as we will show in the next point).

- *$HL(H)$ is closed with respect to passive prefixes.* $HL(H)$ is the set of external traces of all the quiescent executions of H. By contradiction, suppose there is $s \in Pas(HL(H))$ which does not come from a quiescent execution of H. Then there is an extension of this execution which, after a sequence of internal firings, lets an output transition $\bar{o}$ fire. If this is still not quiescent we can do the same thing over again. Note however that H only contains a finite number of external ports and could not continue to output indefinitely, eventually it shall stop and wait for an input, thus reaching a quiescent execution. Then $s \cdot \bar{o} \in HL(H)^{\leq}$ and s is not passive (contradiction).
- *$HL(H)$ is reorder-closed.* Reorder-closedness comes as a consequence of the fact that no output transition may block any other transition but its complement and no input transition may block another input transition. (We say that t *blocks* t' in H just when H contains a path from t to t', where each place has at most one incoming arc.)
- *$HL(H)$ is receptive.* Handshake DB nets are unsafe, that means that places may contain an unlimited number of tokens. So, every time an input transition is enabled to fire it can. Note also that the enabling of an input transition depends only on the alternation with its complement output transition. Then $HL(H)$ is receptive.

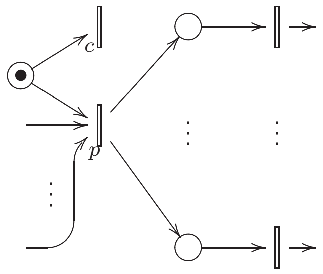
$\square$

Proof of Proposition 4.3 We set up H_σ's external structure by providing both an input and an output transition for each port $p \in P$ and by pairing them together by means of a port structure, as showed in definition 3.1. In particular, the choice of an active or of a passive port structure is taken according to the label $d(p)$. Then we can already state $HS(H_\sigma) = \langle P, d \rangle$. Note also that we have a specific external transition for each message in the alphabet.

Now the internal structure. The occurrence selecters defined in Section 3.2 allow us to associate a new (internal) transition to each occurrence of message: we use specific selecters for inputs as for outputs. The next step is to associate a transition to each position. Recall that a position can be represented as a set containing the last input occurrence of each thread. Then we take a new transition and we link each transition associated to any of these input occurrences into it: the link is a direct arc-place-arc one. We also add a transition for each choice c allowed at a given position $[s]_\sim$. In particular, if c does not stand for the choice to do nothing, we link $[s]_\sim$'s transition to c's transition, again by a direct arc-place-arc link. Note however that c might be in mutual exclusion with another choice c' at $[s]_\sim$, then we need a shared precondition before the corresponding transitions. But the choice of which one to fire should be made once and for all, then this same precondition should be used in any position where the two choices are allowed and mutually exclusive(we draw several outgoing arcs from each choice to mean this):



A special treatment is reserved for the do-nothing choice. In this case we add a transition with no outgoing arcs and put it in mutual exclusion directly with p's transition:



When the choice to do nothing is taken, H_σ has to wait for another input (thus moving to a new position) before doing anything else.

Let us now move to the output side, where each output occurrence $\bar{p}_i$ might be enabled in several positions. Then we make the arcs coming from the choice of $\bar{p}_i$ at each of these positions converge into a unique place, which will have an outgoing arc towards $\bar{p}_i$'s transition. One might object that so doing the same output occurrence might fire twice. But this is prevented by the selecter structure (Section 3.2) which ensures that each transition associated to an occurrence of $\bar{p}$ may fire at most once.

The construction is finally complete, now we prove by induction that $s \in \sigma^{\leq}$ if and only if $s \in HL(H_\sigma)^{\leq}$.

- $s = \varepsilon$. Trivial since both σ and $HL(H_\sigma)$ are handshake languages.
- $s = s' \cdot a$. Let a be an input. Both σ and $HL(H_\sigma)$ are handshake languages on the same handshake structure $\langle P, d\rangle$. Then any direction we look, $s'a$ must be a handshake trace on $\langle P, d\rangle$. Since s' is a prefix of both languages, $s'a$ is too (receptivity). Now let a be the ith occurrence of output $\bar{p}$. $s\bar{p} \in \sigma^{\leq}$ means that $\bar{p}_i$ is allowed by the position $[s]_\sim$ in σ and that no mutually exclusive choice has

been chosen yet. Then in H_σ, $[s]_\sim$'s transition enables the transition associated to the choice of $\bar{p}_i$ at $[s]_\sim$. Plus, if ever there was a shared precondition among the choice of $\bar{p}_i$ and another choice at $[s]_\sim$, we may assume that it still contains a mark since the other choice has not produced any effect so far. Then $\bar{p}_i$'s transition may fire because it has not yet fired and because all the transitions associated to previous occurrences of $\bar{p}$ have already fired in s. Then $s\bar{p} \in HL(H_\sigma)^{\leq}$. On the other hand, $s\bar{p} \in HL(H_\sigma)^{\leq}$ implies that the transition associated to the choice of $\bar{p}_i$ at position $[s]_\sim$ is enabled by the transition associated to $[s]_\sim$ in H_σ. Then $[s]_\sim$ allows $\bar{p}_i$ in σ, by definition of H_σ. Moreover, if there was another choice excluding $\bar{p}_i$ at $[s]_\sim$ in σ, this was not chosen inside s. Then $s\bar{p} \in \sigma^{\leq}$.

Now, s is passive in σ if and only if the transition $[s]_\sim$ does not have any outgoing arcs in H_σ, that is if and only if s is passive in $HL(H_\sigma)$. s is a non-passive trace in σ if and only if the transition $[s]_\sim$ has a shared precondition with a transition which has no outgoing arcs in H_σ, that is if and only if s is a non-passive trace in $HL(H_\sigma)$. $\square$

Proof of Theorem 4.6 The proof of the completeness part of the theorem is a simplification of the proof of Proposition 4.3[4]. As for the the soundness part, the only properties left to prove are determinism's two, given that we already proved the preliminary properties for Theorem 4.1. For both of them, the proof is based on the following simple observation. In a DB net, even if a place may have several outgoing arcs, only one of its post-transitions is actually enabled at any given time: each one has a guard and only one guard contains a mark in the initial state; successively, the firing of the enabled post-transition takes away a mark from its guard and puts it into another guard. This prevents any situation of confusion, so that once a transition is enabled it will stay enabled until it fires.

Also, given two executions ex' and ex'', we can define an execution ex which completes ex' with those firings which occur in ex'' and not in ex' itself. We show how to do this by providing a constructive algorithm which gradually deletes the two original strings ex' and ex'' while writing ex. We initially set ex to the empty string. If at a given time $ex'' = a \cdot u''$ and $ex' = u' \cdot a \cdot v'$, where a does not appear in u', we append a to ex while removing it from the two original strings. So that ex' becomes $u'v'$ and ex'' becomes u''. If a does not appear in ex' we remove it from ex'' and we append it at the end of ex'. If $ex'' = \varepsilon$, we append what is left of ex' at the end of ex. Eventually, ex will consist of all the firings of ex' (possibly in a different order) followed by the firings of ex'' that were not there in ex'. About the order of the firings, note that if ex' and ex'' had the same external trace, ex would still have that external trace.

Now, let $s\bar{p} \in HL(H)^{\leq}$. Recall that $s\bar{p}$ is the prefix of an external trace of a quiescent execution of H. Then $s\bar{p}$ is also the external trace of a prefix of a quiescent execution, then the external trace of an execution of H. For the first property we need to prove that there is no quiescent execution of H whose external trace is s. Since $s \cdot \bar{p} \in HL(H)^{\leq}$, there is an execution ex of H whose external trace is s and which can be extended by $\bar{p}$. Then any execution ex' of H whose external trace is s

[4] Recall also that all deterministic handshake languages are positional (Prop. 2.4).

can be completed with those firings which occur in $ex \cdot \bar{p}$ and not in ex'. Note that the external trace of the execution we obtain is $s \cdot \bar{p}$. Then no execution of H whose external trace is s is quiescent. For the second property, let $s\bar{p}, s\bar{q} \in HL(H)^{\leq}$. Let also $ex'\bar{p}$ be an execution whose external trace is $s\bar{p}$ and $ex''\bar{q}$ be an execution whose external trace is $s\bar{q}$. Just as above we can interleave $ex'\bar{p}$ and $ex''\bar{q}$, so to obtain execution ex whose external trace is $s\bar{p}\bar{q} \in HL(H)^{\leq}$. $\square$

Proof of Lemma 4.5 The direction right-to-left is almost immediate: if there exists such a critical set, by definition $t \nrightarrow_\sigma c$. Then suppose that $t \nrightarrow_\sigma c$. Let S be the set of all critical pairs for c in σ which are inverted in t and let s be a handshake trace with the same threads as t[5] and in which all pairs of S are inverted. We prove by induction on $d = d_\sim(s, t)$ that $s \nrightarrow_\sigma c$:

- $d = 0$. $s = t$, then $s \nrightarrow_\sigma c$;
- $d = l + 1$. There is a sequence $t = t_0, \ldots t_{l+1} = s$ such that $d_\sim(t_i, t_{i+1}) = 1$ and $t_i \nrightarrow_\sigma c$, $\forall 0 \leq i \leq l$. By contradiction assume $t_{l+1} \rightarrow_\sigma c$. Let $t_l = t' \cdot m \cdot n \cdot t''$ and $t_{l+1} = t' \cdot n \cdot m \cdot t''$. If m is an input or n an output, $t_l \ \mathbf{r} \ t_{l+1}$. Then since $t_{l+1} \rightarrow_\sigma c$, also $t_l \rightarrow_\sigma c$. Contradiction. Then m is an output, say the jth occurrence of $\bar{b}$, and n is an input, say the ith occurrence of a. By definition $\langle a_i, \bar{b}_j \rangle$ is a critical pair and since the sequence $t_0, \ldots t_{l+1}$ is minimal by definition of $d_\sim$, $\langle a_i, \bar{b}_j \rangle \in S$. But $\langle a_i, \bar{b}_j \rangle$ is not inverted in s: contradiction!

Then S is critical for c in σ. If S is not minimal we just need to take the minimal critical set contained in S and we are done. $\square$

Proof of Theorem 4.2 We already described the general construction of H_σ for σ positional (proof of proposition 4.3) as well as its extension to the non-positional case (end of section 4). Lemma 4.5 justifies this extended construction by telling us that an "exception" to positionality has all the pairs of a critical set inverted and, viceversa, if a trace has all the pairs of a critical set inverted, then it is an exception to positionality. Then the proofs that $s \in \sigma^{\leq} \iff s \in HL(H_\sigma)^{\leq}$ and $s \in \sigma \iff s \in HL(H_\sigma)$ are just adaptations of the corresponding proofs that we gave for proposition 4.3. $\square$

[5] To be more precise we should take s from a larger set, where the number of input occurrences in each thread of s is equal to the number of input occurrences in the corresponding thread of t. This allows a thread of s to differ from the corresponding thread of t by an output occurrence. However reorder-closedness implies that outputs do not affect choices, so that we can assume s and t have exactly the same threads.