Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 171 (2007) 21–41

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

# Handshake Games

Luca Fossati[1] ,[2]

*Dipartimento di Informatica*
*Università di Torino*
*Torino, Italia*

**Abstract**

In this paper I present a game model for the semantical analysis of handshake circuits. I show how the model captures effectively the composition of circuits in an associative way. Then I build a compact-closed category of handshake games and handshake strategies. I then consider the language Tangram and I define a semantics for this language simply by giving a denotation in the model to each handshake component that is used in the compilation of Tangram programs.

*Keywords:* Game semantics, handshake circuits, nondeterminism, concurrency, associativity.

## 1 Introduction

The handshake protocol has experienced a great commercial success as a paradigm for asynchronous communication and computation. It is a protocol of communication between circuits, which are connected through *channels* over which they exchange information. In particular a circuit sends a message over a channel through an interface called *port.* In a communication over a single channel, one circuit takes the active role and sends the first message while the other is initially waiting for an activation sign. Then the former waits for the latter to reply, and so on. From a circuit's point of view, this behavior induces an alternating sequence of input and output messages, *requests* and *acknowledges.*

The handshake technology has been employed in parts of several integrated systems and in the Philips Research Labs Van Berkel et al. have developed a language called Tangram, a *Hardware Design Language* (HDL), so that hardware design becomes simply a programming task. One reason why this is possible is modularity, with an architecture that does not rely on a central clock. Modularity also helps

[1] I thank Felice Cardone for having started me on the subject, Russ Harmer, Paul-André Melliès and Pierre-Louis Curien for useful hints on the critical part, Simonetta Ronchi Della Rocca for support and directions.

[2] Email: fossati@di.unito.it

doi:10.1016/j.entcs.2006.11.034

these circuits gain in efficiency and speed, a slow module has a lower impact on the overall system. Another language in which programs are compiled into handshake circuits has been developed by the Department of Computer Science at the University of Manchester [6] and is called Balsa. But in contrast with the increase of interest in the implementations of the protocol, the foundational investigation rests still as it was ten years ago. For example, in the semantics of Tangram proposed by Van Berkel data exchange is not taken into account [7], and since then no other semantics has filled this hole. Moreover all the literature relies on Van Berkel's model [28]: indeed, a very natural characterization of circuits with a nice analysis of their behavior; however some points are still not treated in sufficient detail.

The main of these is surely the analysis of composition. The property of *delay insensitivity* allows us to ignore the order in which messages reach destination, to do as if it was the same order in which they have been sent ([28], page 75). Still, it remains not trivial to model composition precisely. Surely handshake circuits compose and surely they compose in the same way no matter the order of compositions (associativity): in the end it is just the same physical phenomenon observed from different angles... but, does the model capture the observation of the physical phenomenon faithfully? Is any quiescent point (inside a communication) taken into account by the model? As expected, the problem rises in presence of the "infamous" *infinite internal chatters* and really causes big troubles, as associativity of composition is a fundamental property.

While looking deeper into the issue I found a counter-example to the associativity of composition in Van Berkel's model. Later on, Russ Harmer pointed out to me another counter-example, due to Roscoe [27], which tells that any model of unbounded nondeterminism must list explicitly the possibly infinite communications inside the process' description. My counter-example is detailed in the Appendix (section 6).

The search for a solution was based on previous game semantics of nondeterminism and concurrency [13][14][4][19][15][22] in which the problem of composition is treated in full detail. The choice is in part motivated by the many similarities between the game paradigm and the handshake protocol. The common view of computation as interaction is the key ingredient, from which follow several other correspondences, where all the dualisms are reflected. In most cases it is just a matter of switching to a new vocabulary:

- $player/opponent \iff system/environment$;
- $negative/positive \iff passive/active$;
- $move \iff message$;
- $P-move/O-move \iff input/output$;
- $query/response \iff request/acknowledge$.

One could also argue that games correspond to handshake structures and strategies to handshake processes [3] but on these aspects the correspondence is not so clear

[3] This would allow one to give a neat definition of handshake circuits as specifications of behaviors over

in the handshake circuits literature. There is no clear distinction between a process and its structure, no concept of a type. This is another motivation for adopting the game formalism: definitely game models rely on powerful mathematical structures such as categories and logic, and these are the perfect tools for defining a type theory. The work could be placed inside Abramsky's program to bridge the historical gap in formal semantics, between the family of functional models (denotational semantics) and models of concurrency (process calculi) ([1] and [2], among other works).

Combining nondeterminism with concurrency and asynchrony has been a major issue to deal with. Moschovakis published several works in that direction (starting from [24]) but I believe the aim there was more directed to the verifications of properties (fairness) than to the structural analysis, which makes his works essentially different from mine. A reasonable choice was to extend the model for finite nondeterminism of Harmer and McCusker [13] but unfortunately that is justified only with the assumption of sequentiality of the overall communication (the strict alternance within plays). So I turned to countable nondeterminism [14], reformulated in a way that allows extension to parallelism. Basically in the new formalism, nondeterministic strategies are seen as sums of deterministic strategies. The idea of such a presentation was given me by Paul-André Melliès during a discussion and follows quite a few works in the literature, starting from [26] and including the works by Moschovakis.

## 2 Notations

In the following I will give a few definitions that always turn out useful when dealing with the formal aspects of concurrency. Some are inherited from Mazurkiewicz's seminal work on trace theory [21], even though I changed them in order to fit them in the current context.

We define a *concurrent alphabet* as an alphabet $M$ equipped with a binary reflexive symmetric and transitive relation $D$ over its letters (moves, as we will call them). $D$ is called the *dependence* relation. Being an equivalence relation, $D$ decomposes the alphabet into equivalence classes which we call *partitions*. A string over $\langle M, D\rangle$ is just a string over $M$. The complementary relation is independence, $I$. Sometimes I might use $D$ and $I$ as relations on strings. In particular, for two strings $s$ and $t$, $s\ D\ t$ if there are two moves $m$ and $n$, appearing in $s$ and $t$ respectively, and such that $m\ D\ n$.

Let $\langle M, D\rangle$ be a concurrent alphabet. Every string $s$ over $M$ has an underlying graph induced by $D$. The vertices are the moves in $s$ and edges between two moves are present if and only if the two moves are dependent with each other. Any maximal clique is the underlying graph of a unique string $t$ contained in $s$, where moves appear in the same order in $t$ as they appear in $s$. We say $t$ is a *thread* of $s$. A thread could also be seen as a string over a partition of $\langle M, D\rangle$. The *initial*

structures.

move of a thread $t$ is the first move that appears in $t$. The initial moves of a string $s$ are the initial moves of each of its threads.

Consider the above concurrent alphabet and let $D' \subseteq D$. $D'$ is a *full subdependence* of $D$ when it preserves reflexivity, transitivity, and symmetry (the dependence relation of a partition is a full subdependence, for example). $m \in M$ is *involved* in $D'$ when $\exists n \in M$ such that $(m\ D'\ n)$. Now let $D'$ be a full subdependence of $D$, $s$ a string over $M$ and $M' \subseteq M$ the set of moves involved in $D'$. We can define $s \restriction \langle M', D' \rangle$ as the string obtained by keeping only the moves from $M'$ in the same order as they appear in $s$. Normally we write just $s \restriction M'$ assuming we have a full subdependence associated to it.

Finally recall the standard order $\sqsubseteq$ on strings $s$ and $t$:

$$s \sqsubseteq t \iff t = st'$$

where $t'$ is a string over $\langle M, D \rangle$. We say that $t$ is an *extension* of $s$ with the moves in $t'$. We write $len(s)$ for the length of a string $s$.

## 3 Handshake Games

*Game.* We can imagine a game as providing a universal structure over which several processes of the same kind can be implemented.

**Definition 3.1** A **handshake game** is a structure $A = \langle M_A, D_A, \lambda_A \rangle$, where $M_A$ is a set of moves and $D_A$ a dependence relation on them. Together they form a concurrent alphabet on which we impose a finite number of partitions. $\lambda_A : M_A \to \{-, +\} \times \{R, A\}$ is a labelling function, we denote with $\lambda_A^{-+}$ and $\lambda_A^{RA}$ its two projections. The first one determines the polarity, the moves with positive polarity are called player moves and those with negative polarity are called opponent moves. The second projection distinguishes *requests* ($R$) from *acknowledges* ($A$). We impose no ambiguity of labels within a partition. Given $m, n \in M_A$ such that $m\ D_A\ n$:

$$\lambda^{-+}(m) = \lambda^{-+}(n) \iff \lambda^{RA}(m) = \lambda^{RA}(n)$$

**Example 3.2** The simplest case is the game[4] associated to a port structure, where all moves are in the same equivalence class, as they represent a set of messages which are all going to be sent over the same channel. For example, we can associate a generic *passive* port to a game $A = \langle M_A = M_1 \cup M_2, D_A, \lambda_A \rangle$ such that:

- $M_1 = \{req(v) \mid v \in V_1\}$ and $M_2 = \{ack(v) \mid v \in V_2\}$;
- $D_A = M_A \times M_A$;
- $\forall m \in M_1, (\lambda^{-+}(m) = -) \wedge (\lambda^{RA}(m) = R)$;
- $\forall m \in M_2, (\lambda^{-+}(m) = +) \wedge (\lambda^{RA}(m) = A)$.

The port is passive, and this finds semantic correspondence in that only the opponent can issue a request, only the opponent is allowed ‘to start’. The sets $V_1$ and $V_2$

[4] From now on we can leave the adjective, handshake, implicit.

represent the sets of values the two players can attach to their moves. Special cases occur when these sets are singletons, or equivalently when no data are added. In particular when there are only one request and one acknowledge we are representing a *nonput* port. *Input* ports are obtained by allowing the opponent to encode data in its moves while in *output* ports it is the player who can encode data in its moves. If they both can, then we have a *biput* port.

*Connectives.* A simple operation allows us to change a game's polarity:

$$A^{\perp} = \langle M_A, D_A, \lambda_{A^{\perp}} \rangle.$$

Where $\lambda_{A^{\perp}} = \langle -\lambda_A^{-+}, \lambda_A^{RA} \rangle$, and $-\lambda_A^{-+}$ gives $-$ when $\lambda_A^{-+}$ gives $+$ and viceversa. So for example, a generic *active* port can be described as the dual $A^{\perp}$ of the generic passive port $A$, described above.

We also have a binary connective, the *ostar product.* This allows us to give a representation to more complex structures, structures with more than one port. Given two games $A$ and $B$ their ostar product is:

$$A \circledast B = \langle M_A + M_B, D_A + D_B, \lambda_A + \lambda_B \rangle$$

Note that $D_A$ and $D_B$ are full subdependencies of $D_{A \circledast B}$ and $M_A$ ($M_B$) is exactly the set of moves involved in $D_A$ ($D_B$). Then we can relax the definition of restriction and write $s \upharpoonright A$ ($s \upharpoonright B$) instead of $s \upharpoonright \langle M_A, D_A \rangle$ ($s \upharpoonright \langle M_B, D_B \rangle$).

*Play.* As usual in games, player and opponent take turn to play the respective moves. Here though we liberalize things a little as we are working in a concurrent framework, we impose the turn alternation only on the *threads* of our play (look at it as if we were playing several games in parallel). Formally a *play*, on a game $\langle M_A, D_A, \lambda_A \rangle$[5] is just a string over the concurrent alphabet $\langle M_A, D_A \rangle$. A play is *legal* if and only if:

- all its initial moves are requests;
- all its threads are alternating sequences of requests and acknowledges, player and opponent moves.

We write $\mathcal{L}_A$ for the set of legal plays over the game $A$. We write $\mathcal{L}_A^{fin}$ and $\mathcal{L}_A^{\omega}$ for the subsets of finite and infinite legal plays, respectively. Note that a play over a game can only contain a finite number of threads (finite concurrency) because the alphabet of the game can only have a finite number of partitions. The equivalence on moves ($D_A$) induces an equivalence on plays, the *homotopy* relation $\sim_A$. Let $s, t \in \mathcal{L}_A$, we say that $s \sim_A t$ just when they have the same set of threads[6].

*Prestrategy.* I will introduce strategies gradually: starting from the general class of prestrategies I will characterize positionality and determinism; by then we will be working with actual strategies but in order to consider nondeterministic behaviors as

[5] The term play is inherited from game semantics. It corresponds to the handshake traces in the handshake circuits theory.

[6] The use of homotopy in concurrent games is due to Melliès and Mimram [22][23].

well we will need a new generalization. All of these classes really need the adjective *handshake* as we are working in a handshake framework, but since it will be clear from the context I will leave it implicit. I start with some formal notions. Given a set of plays $\sigma$ on a game $A$:

- The prefix-closure of $\sigma$ is

$$\sigma^{\leq} = \{s \in \mathcal{L}_A \mid s \sqsubseteq t \in \sigma\}$$

- For a play $s \in \sigma^{\leq}$, its *successors set* (with respect to $\sigma$) is:

$$suc \cdot (s, \sigma) = \{m \in M_A \mid sm \in \sigma^{\leq}\}$$

- $s$ is *passive* in $\sigma$, $pas \cdot (s, \sigma)$, iff there is no move the player can make at $s$:

$$m \in suc \cdot (s, \sigma) \Rightarrow \lambda^{-+}(m) = -$$

- $Pas \cdot (\sigma)$ is the set of passive plays in the prefix-closure of $\sigma$:

$$Pas \cdot (\sigma) = \{s \in \sigma^{\leq} \mid pas \cdot (s, \sigma)\}$$

- Given two independent moves, $m, n \in M_A$, and four plays $r$, $s$, $t$, $u$, of $A$, we define $\mathbf{r}_A$ as the smallest binary relation such that:
  · $(mn\,\mathbf{r}_A\,nm) \iff ((\lambda^{-+}(m) = -) \vee (\lambda^{-+}(m) = \lambda^{-+}(n)))$[7];
  · $t\,\mathbf{r}_A\,t$;
  · $(r\,\mathbf{r}_A\,s) \wedge (s\,\mathbf{r}_A\,t) \Rightarrow (r\,\mathbf{r}_A\,t)$;
  · $(r\,\mathbf{r}_A\,s) \wedge (t\,\mathbf{r}_A\,u) \Rightarrow (rt\,\mathbf{r}_A\,su)$.
  We say that $s$ *reorders* $t$ in $A$ when $s\,\mathbf{r}_A\,t$[8];
- Let $s$, $t$ be two plays of $A$ and define $s\,\mathbf{x}_A\,t$ if and only if $s \sqsubseteq t$ and $t$ contains only opponent moves after $s$ (we say that $t$ is an *input-extension* of $s$ in $A$).

**Definition 3.3** A *(handshake) prestrategy* $\sigma$ on a game $A$ is a set of legal plays of $A$ such that:

(i) $\sigma \neq \emptyset$ (non-empty);

(ii) $Pas \cdot (\sigma) \sqsubseteq \sigma$ (closed under passive prefixes);

(iii) $(t \in \sigma \wedge s\,\mathbf{r}_A\,t) \Rightarrow s \in \sigma$ (reorder closed);

(iv) $(s \in \sigma^{\leq} \wedge s\,\mathbf{x}_A\,s') \Rightarrow s' \in \sigma^{\leq}$ (receptive).

$\sigma$ is conveniently described as a couple $\langle \mathcal{Q}_\sigma, \mathcal{I}_\sigma \rangle$. Where $\mathcal{Q}_\sigma$ is called the set of *quiescent* and $\mathcal{I}_\sigma$ the set of *infinite* plays of $\sigma$.

The explicit inclusion of infinite plays allows to distinguish when a strategy may follow an infinite communication but at a certain point will surely stop and when

[7] Let me give one intuition here ... We may receive two inputs in any order and we may output two messages in any order. Also, an input may arrive before we output. The converse does not hold though, we may have to wait for an input before we can output.

[8] I chose to keep the definition as in the original model instead of extending it to infinite sequences of reorderings, this is coherent with the degree of 'intensionality' in the model

instead it could continue to the infinite. The quiescent plays represent the finite points in which the player may stop to wait for the opponent to move. The passive plays represent those points at which the player has no other choice but to wait: they need to be quiescent. The notions of reorder-closedness and receptivity are inherited from the handshake circuits literature, but curiously I discovered then that they had already been introduced to the game semantics community. Ghica and Murawski in their works on concurrent games for model-checking consider *O-complete* (receptive) and *saturated* (reorder-closed) strategies (look for example at [11]). Also, Mimram and Melliès [23] have an ongoing work on games in the more general context of diagrams and there they introduce a constraint called *courtesy* which seems to correspond to reorder-closedness.

In the case of nonput ports, some well known prestrategies are $STOP$ and $SKIP$. $STOP$ simply accepts opponent moves without acknoledging them, while $SKIP$ does acknowledge to requests, but never starts a handshake itself:

- $STOP_A = \{s \in \mathcal{L}_A \mid (s)_+ = 0\}$;
- $SKIP_A = \{s \in \mathcal{L}_A \mid (s)_- = (s)_+\}$.

Where $(s)_-$ and $(s)_+$ stand for the number of opponent and that of player moves in $s$, respectively. It is interesting to see that if we add data to moves, a prestrategy like $SKIP$ becomes a highly nondeterministic specification after which a wide range of possible prestrategies could be implemented. Let's consider the simple case where $A$ represents a single passive port. If data were added to requests no nondeterminism would be introduced. On the other hand if only booleans were added to acknowledges, the player would have many available options. It could always answer $tt$ or $ff$, or alternate the two answers, ... Even more interesting if the port is a biput. Here the number of possible implementations is extremely high: identity, boolean not, constant, integer successor, test for zero, ...

*Composition.* Given two prestrategies, $\sigma$ on $A^\perp \otimes B$ and $\tau$ on $B^\perp \otimes C$, their composition is soon defined:

$$\tau \circ \sigma = \{u \upharpoonright A, C \mid u \in \mathcal{L}_{A,B,C} \wedge u \upharpoonright A, B \in \sigma \wedge u \upharpoonright B, C \in \tau\}$$

Two definitions related to composition. $s \in \mathcal{L}^{fin}_{A,B,C}$ is an *interleaving* in the composition of $\sigma$ and $\tau$ if and only if $s \upharpoonright A, B \in \sigma^{\leq}$ and $s \upharpoonright B, C \in \tau^{\leq}$. $s$ is a *witness* (for $s \upharpoonright A, C$) in the composition of $\sigma$ and $\tau$ if and only if $s \upharpoonright A, B \in \sigma$ and $s \upharpoonright B, C \in \tau$.

Unfortunately, composition of prestrategies is not well-defined. Informally: consider the general prestrategy $RUN$ which is always eager to get engaged in a handshake, whether its role is to start or to answer. With abuse of notation we could say that the prestrategy $RUN$ on a game $B$ contains all and only those plays which are passive with respect to $B$. Consider the composition of $RUN_B$ with $RUN_{B^\perp}$. Being dual processes they will never agree on being quiescent at the same time, each one will be willing to continue (or to start over) immediately when its turn comes. Then if the infinite play which emerges from this eternal ping-pong is not itself contained in the two prestrategies, the composition is empty. Somehow we need to force certain infinite plays to be included in the strategies.

*Positionality.* We need to focus on a subset of the set of prestrategies for which composition works. It seems natural then to start with deterministic prestrategies. However it turns out that determinism is preserved by composition only in presence of another very important property of concurrent strategies, *positionality.* Positionality is based on the relation of homotopy between plays.

**Definition 3.4** A prestrategy $\sigma$ on a game $A$ is *positional* if and only if, for finite $s, s' \in \sigma^{\leq}$ and infinite $t, t' \in \mathcal{L}^{\omega}_{A}$, with $s \sim s'$ and $t \sim t'$, we have:

(i) $s \cdot \bar{s} \in \sigma^{\leq} \Rightarrow s' \cdot \bar{s} \in \sigma^{\leq}$ (positionality and prefixes);

(ii) $s \in \sigma \Rightarrow s' \in \sigma$ (positionality and quiescence);

(iii) $t \in \sigma \wedge (\forall \bar{t} \sqsubset t', \bar{t} \in \sigma^{\leq}) \Rightarrow t' \in \sigma$ (positionality and infinite plays);

Intuitively, a position is a state and we can reach a certain state in different ways but then however we got there we have the same options to move on or to wait.

*Determinism.* Let $A$ be a game and $a$ and $b$ two distinct player moves of $A$. A positional prestrategy $\sigma$ on $A$ is *deterministic* just when:

(i) $ta \in \sigma^{\leq} \Rightarrow t \notin \sigma$;

(ii) $ta \in \sigma^{\leq} \wedge tb \in \sigma^{\leq} \Rightarrow tab \in \sigma^{\leq}$;

(iii) $\mathcal{I}_{\sigma} = \{t \in \mathcal{L}^{\omega}_{A} \mid \forall t' \sqsubset t, m \in suc \cdot (t', \sigma),\ \lambda^{-+}(m) = +, \exists t'' \sqsubset t \text{ s.t. } t' \sqsubset t'' \cdot m \sqsubset t\}$.

The first two conditions are the usual conditions for determinism. Moreover a deterministic strategy that has engaged in an infinite chatter may not decide to quit anyhow. A malicious opponent could force a deterministic strategy to follow the course that he wants, even to diverge. If the opponent acts in this way there is no means for the deterministic strategy to escape the malicious design. The third condition tells exactly which are the infinite plays that a deterministic strategy must contain.

It's clear that the third condition of determinism implies the third condition of positionality. So for deterministic prestrategies positionality can be expressed with two properties.

*Composition.* We now proceed to the proof that composition is well-defined for deterministic positional prestrategies. In the following let $\sigma$ and $\tau$ be two deterministic positional prestrategies on $A^{\perp} \otimes B$ and $B^{\perp} \otimes C$, respectively. We start with two definitions.

Given $u$ and $v$ such that $u\ \mathbf{r}\ v$, we define $d_{\mathbf{r}}\ (u, v)$, the *reordering distance* between $u$ and $v$:

- $d_{\mathbf{r}}\ (u, v) = 0 \iff u = v$
- $d_{\mathbf{r}}\ (u, v) = 1 \iff (u = u' \cdot a \cdot b \cdot u'') \wedge (v = u' \cdot b \cdot a \cdot u'')$, for two moves $a$ and $b$;
- In general $d_{\mathbf{r}}\ (u, v) = n > 0$ if and only if $u \neq v$ and there are $n + 1$ (and no less) plays $u_0, u_1, \ldots u_n$ such that $u = u_0$, $u_n = v$ and $d_{\mathbf{r}}\ (u_i, u_{i+1}) = 1$.

In exactly the same way we define the homotopy distance $d_\sim(u, v)$ on homotopic plays $u \sim v$.

**Lemma 3.5** *Let $s \in \mathcal{L}^{fin}_{A,B,C}$ be an interleaving in the composition of $\sigma$ and $\tau$. Then there is $t \sqsupseteq s$ and $t$ is a witness in the composition of $\sigma$ and $\tau$.*

**Proof.** If $s \upharpoonright A, B \in \sigma$ and $s \upharpoonright B, C \in \tau$, $s$ itself is the witness. Otherwise, suppose $\sigma$ is not quiescent at $s$. Then $s$ is not passive in $\sigma$, $\sigma$ can play a move after $s$. We start by saturating the external threads of $\sigma$ with player moves. Then we let $\sigma$ play in $B$ until quiescence, while $\tau$ must be ready to accept this stream (receptivity). Now, if $\tau$ is quiescent we are done, otherwise we saturate it outside. If it becomes quiescent we are done as well otherwise we let it play inside until quiescence. Then it's $\sigma$'s turn again, and so on. If after a while they both become quiescent then we have our witness. Otherwise they run to the infinite, but then we also have our witness (determinism, third condition). □

**Lemma 3.6** *Let $t \in (\tau \circ \sigma)^{\leq}$ and let $u$ and $v$ be two interleavings for $t$ in $\tau \circ \sigma$. Then $u$ can be completed with all the occurrences of moves that are in $v \setminus u$ so to obtain an interleaving $z$ for $t$ in $\tau \circ \sigma$.*

**Proof.** A few remarks. After $\sigma$ ($\tau$) makes a move $m$ it can still play all the moves it could play before but $m$ itself (determinism, second condition). $\tau$ ($\sigma$) instead can still play all the moves it could play before (receptivity and reordering) and possibly more. It follows that if $m$ is initial in $v$ and $m\ I\ u$ then $u' \cdot m \cdot u''$ is still an interleaving for $t$ in $\tau \circ \sigma$, for any factorization $u' \cdot u''$ of $u$. Else if $m\ D\ u$ then $u = u' \cdot n \cdot u''$, where $m\ I\ u'$ and $m\ D\ n$. Then both $m$ and $n$ could be played after $u'$ and they would extend the same thread, since they are dependent. Then they have the same polarity (definition of thread). Then $m = n$ since both $\sigma$ and $\tau$ are deterministic.

We can obtain $z$ as the result of calling the following recursive algorithm on $u$ and $v$ (the language is informal and untyped, nonetheless it should be pretty clear in my opinion):

```
union($u, v$){
  $m$ := head($u$);
  $\bar{u}$ := tail($u$);
  if ($m\ I\ v$) then
    if ($\bar{u} = \varepsilon$) then $z := m \cdot v$;
    else $z := m \cdot$ union($\bar{u}, v$);
  else if ($v = v' \cdot m \cdot v''$) then
    $z := m \cdot$ union($\bar{u}, v' \cdot v''$);
  return $z$;
}
```

□

**Theorem 3.7** *Let $\sigma$ and $\tau$ be two deterministic positional prestrategies on $A^\perp \otimes B$ and $B^\perp \otimes C$, respectively. Then,*

*pre)* *$\tau \circ \sigma$ is a prestrategy;*

*pos)* *$\tau \circ \sigma$ is positional;*

*det)* *$\tau \circ \sigma$ is deterministic.*

**Proof.**

*pre)* I skip the proof that every play in $\tau \circ \sigma$ is legal as it is almost immediate.

(i) $\tau \circ \sigma$ is non-empty. Since neither $\sigma$ nor $\tau$ is empty, then $\varepsilon$ is an interleaving of their composition. Then lemma 3.5 applies and yields a witness $t \sqsupseteq \varepsilon$ for some play in $\tau \circ \sigma$;

(ii) Let $s \sqsubseteq \bar{s}$, with $\bar{s} \in \tau \circ \sigma$ and $s \in Pas \cdot (\tau \circ \sigma)$. By definition there is $\bar{t} \in \mathcal{L}_{A,B,C}$ such that $\bar{t} \restriction A, B \in \sigma$, $\bar{t} \restriction B, C \in \tau$ and $\bar{t} \restriction A, C = \bar{s}$. Just cut $\bar{t}$ right after $s$ is played out in the external game and call the resulting play $t$. Then lemma 3.5 applies and yields a witness $t' \sqsupseteq t$ for some play in $\tau \circ \sigma$. $s$ is passive then $t'$ is actually a witness for $s$[9];

(iii) Let $s \, \mathbf{r}_{A^\perp, C} \, \bar{s}$. Consider $d = d_{\mathbf{r}} \, (s, \bar{s})$:

$d = 0$ Then there is a witness $\bar{u}$ of $\bar{s}$ which is also a witness of $s$;

$d = n$ Then there are $s_0, s_1, \ldots s_n \in \mathcal{L}_{A,C}$ and such that $s = s_0, s_n = \bar{s}, d_{\mathbf{r}} \, (s_i, s_{i+1}) = 1$ for all $0 \leq i \leq n$. For assumption we know that $s_1 \in (\tau \circ \sigma)$. Let $s_1 = s' \cdot m \cdot n \cdot s''$ and $s_0 = s' \cdot n \cdot m \cdot s''$. Moreover let's take a witness $u_1 \in \mathcal{L}_{A,B,C}$ of $s_1$ of the form $u_1 = u' \cdot m \cdot u_b \cdot n \cdot u''$, where $u'$ and $u''$ are interleavings of $s'$ and $s''$, respectively, and $u_b$ is a sequence of moves in $B$. If $n$ is an opponent move then we define $u_0 = u' \cdot n \cdot m \cdot u_b \cdot u''$ else $m$ is a player move and then we define $u_0 = u' \cdot u_b \cdot n \cdot m \cdot u''$. In any case $u_0 \restriction A, C = s$, $u_0 \restriction A, B \, \mathbf{r}_{A^\perp, B} \, u_1 \restriction A, B \in \sigma$ and $u_0 \restriction B, C \, \mathbf{r}_{B^\perp, C} \, u_1 \restriction B, C \in \tau$, then $s \in \tau \circ \sigma$.

(iv) Let $s \in (\tau \circ \sigma)^{\leq}$ and let $s \, \mathbf{x} \, s \cdot s'$. Reasoning as in the second point we can find an interleaving $u$ for $s$ in the composition $\tau \circ \sigma$. It follows from receptiveness of the two prestrategies that $u \cdot s'$ is also an interleaving. Then lemma 3.5 applies and we are done.

*pos)* As already remarked, the third property of positionality follows from determinism which will be proved later on. The first two properties can be proved in exactly the same way, so I prove just the first one. Let $s, s' \in (\tau \circ \sigma)^{\leq}$, with $s \sim s'$, and let $s \cdot \bar{s} \in (\tau \circ \sigma)^{\leq}$. The proof that $s' \cdot \bar{s} \in (\tau \circ \sigma)^{\leq}$ is by induction on $d_\sim \, (s, s')$:

- If $s = s'$ then trivially $s' \cdot \bar{s} \in (\tau \circ \sigma)^{\leq}$;
- If $d_\sim \, (s, s') = n+1$ then there is $s_n$ such that $d_\sim \, (s, s_n) = n$ and $d_\sim \, (s_n, s') = 1$. By inductive hypothesis $s_n \cdot \bar{s} \in (\tau \circ \sigma)^{\leq}$. Now, $s_n$ factorizes as $t' \cdot a \cdot b \cdot t''$ while $s' = t' \cdot b \cdot a \cdot t''$. So $s_n$ has an interleaving $u_n = u' \cdot a \cdot u_b \cdot b \cdot u''$, where $u_b$ is a sequence of internal moves. The two prestrategies are deterministic then it follows from the remarks made inside lemma 3.6 that $u' \cdot b \cdot a \cdot u_b \cdot u''$ is also an interleaving of $\tau \circ \sigma$ (in particular if $b$ is by the opponent, receptivity also plays

[9] Note that in the proof of the lemma no external opponent move is taken while making $t'$, then really no move is played outside.

a role). Finally by positionality of $\sigma$ and $\tau$ we conclude $s' \cdot \bar{s} \in (\tau \circ \sigma)^{\leq}$.

*det*)(i) Let $t \cdot a \in (\tau \circ \sigma)^{\leq}$, where $a$ is a player move. For contradiction, $t \in \tau \circ \sigma$. Then $t$ has a witness $u \in \mathcal{L}_{A,B,C}$. Consider the case where $u$ is finite. Since $u \upharpoonright A, B \in \sigma$ and $u \upharpoonright B, C \in \tau$, neither prestrategy can play at this point, the next move must be an opponent move in $A$ or $C$, and this is enough for the first case. If $u$ is infinite then any move that could be played after a finite prefix will eventually be played somewhere in $u$, including $a$;

(ii) Let $s \cdot a, s \cdot b \in (\tau \circ \sigma)^{\leq}$, where $a$ and $b$ are two distinct player moves. An interleaving for $s \cdot a$ is of the form $u' \cdot a \cdot v'$ and an interleaving for $s \cdot b$ is of the form $u'' \cdot b \cdot v''$. Where $u'$ and $u''$ are both interleavings for $s$ and $v'$ and $v''$ are sequences of internal moves. Then we can apply lemma 3.6 and complete $u'$ with the moves that are not in it but in $u''$. In the same way we can complete $u''$ with the moves that are not in it but in $u'$. The two results are homotopic. Take one of them, $u$. By the remarks made inside lemma 3.6 and by positionality we have that $u \cdot a$ is both an interleaving of $s \cdot a$ and of $s \cdot b$, then the conclusion follows;

(iii) This is actually a double implication. Let $s \in \mathcal{L}^{\omega}_{A,C}$ such that all its prefixes are in the prefix-closure of $\tau \circ \sigma$:

$\Rightarrow$) Suppose $s \in (\tau \circ \sigma)$ and $m \in suc \cdot (s', \tau \circ \sigma)$, where $s' \sqsubset s$ and such that $\lambda^{-+}(m) = +$. Then $s$ has a witness $u \in \mathcal{L}^{\omega}_{A,B,C}$. If either $u \upharpoonright A, B$ or $u \upharpoonright B, C$ is finite then respectively $\sigma$ and $\tau$ cannot move anymore. That means that if it was $\sigma$'s ($\tau$'s) duty to play (the occurrence of) $m$ then the move occurs in the finite restriction (first condition of determinism). Suppose that $m$ could be played on the side of the communication which is infinite. Then again we know that the move will eventually occur (third condition of determinism);

$\Leftarrow$) Suppose that forall $s' \sqsubset s$ and $m \in suc \cdot (s', \tau \circ \sigma)$ with $\lambda^{-+}(m) = +$ there exists $s'' \sqsubset s$ such that $s' \sqsubset s'' \cdot m \sqsubset s$. Take $u \in \mathcal{L}^{\omega}_{A,B,C}$ such that all the prefixes of $u \upharpoonright A, B$ are in the prefix-closure of $\sigma$, all the prefixes of $u \upharpoonright B, C$ are in the prefix-closure of $\tau$ and $u \upharpoonright A, C = s$. For any $u' \sqsubset u$ we know that if either prestrategy could play an external move after $u'$ then this eventually occurs in $u$, because the same move could be played by $\tau \circ \sigma$ after the external restriction of $u'$. If either prestrategy could play an internal move and the (occurrence of the) move does not appear in $u$ yet, then we add it, say, ten moves after $u'$. This way we are assured that every move that could be played after a finite prefix actually occurs inside a finite prefix[10]. The final $\bar{u}$ is actually a witness for $s$ in $\tau \circ \sigma$.

□

*Sums and Strategies.* We define the binary operation $\oplus$ (*sum*) which takes two prestrategies $\sigma$ and $\tau$ on a game $A$ and returns the union of their sets of plays.

**Lemma 3.8** *The sum of $\sigma$ and $\tau$ is again a prestrategy.*

[10] Note that the number of moves that can be played at any point is finite, as the number of threads is finite (finite concurrency) and the two prestrategies are deterministic.

**Proof.**

(i) The union of two non-empty sets is non-empty.

(ii)

$$\begin{aligned} s \in Pas \cdot (\sigma \oplus \tau) &\Rightarrow s \in (\sigma \oplus \tau)^{\leq} \wedge pas \cdot (s, \sigma \oplus \tau) \\ &\Rightarrow (s \in \sigma^{\leq} \wedge pas \cdot (s, \sigma)) \vee (s \in \tau^{\leq} \wedge pas \cdot (s, \tau)) \\ &\Rightarrow (s \in Pas \cdot \sigma) \vee (s \in Pas \cdot \tau) \\ &\Rightarrow (s \in \sigma) \vee (s \in \tau) \\ &\Rightarrow s \in \sigma \oplus \tau \end{aligned}$$

(iii) Reorder-closedness and receptivity can be proven similarly, so we prove only the first one of the two. Let $t \in (\sigma \oplus \tau) \wedge s\, \mathbf{r}_A\, t$:

$$\begin{aligned} t \in \sigma \oplus \tau &\Rightarrow (t \in \sigma) \vee (t \in \tau) \\ t \in \sigma \wedge s\, \mathbf{r}_A\, t &\Rightarrow s \in \sigma \\ t \in \tau \wedge s\, \mathbf{r}_A\, t &\Rightarrow s \in \tau \\ (s \in \sigma) \vee (s \in \tau) &\Rightarrow s \in \sigma \oplus \tau \end{aligned}$$

□

We extend this operation to an arbitrary number of arguments in the expected way, then we are ready for strategies.

**Definition 3.9** A *(handshake) strategy* $\sigma$ is the sum of deterministic positional (handshake) prestrategies:

$$\sigma = \bigoplus_{i \in I} \sigma_i$$

where $\sigma_i$ are deterministic positional prestrategies indexed by elements of a non-empty set $I$.

A strategy represents a process. A strategy is a description of the behavior to follow. It is as if strategies could choose once and for all among a set of possible behaviors. Afterwards they act deterministically all the way. It follows that a deterministic positional prestrategy $\sigma_d$ is just a particular case of strategy where the opponent may choose only from a singleton set. We will just say that $\sigma_d$ is a deterministic strategy, assuming implicitly that it is positional, even if strategies are not positional in general.

**Lemma 3.10** *Given two strategies $\sigma$ and $\tau$ on a game $A$, their composition is again a strategy.*

**Proof.** Let

$$\sigma = \bigoplus_{i \in I} \sigma_i \qquad\qquad \tau = \bigoplus_{j \in J} \tau_j$$

where all $\sigma_i$'s and $\tau_j$'s are deterministic prestrategies. Then we have:

$$\begin{aligned}\tau \circ \sigma &= \{u \upharpoonright A, C \mid u \in \mathcal{L}_{A,B,C} \wedge u \upharpoonright A, B \in \sigma \wedge u \upharpoonright B, C \in \tau\} = \\ &= \{u \upharpoonright A, C \mid u \in \mathcal{L}_{A,B,C} \wedge u \upharpoonright A, B \in \oplus_{i \in I}\sigma_i \wedge u \upharpoonright B, C \in \oplus_{j \in J}\tau_j\} = \\ &= \{u \upharpoonright A, C \mid u \in \mathcal{L}_{A,B,C} \wedge (\exists i \in I, j \in J \text{ s.t. } u \upharpoonright A, B \in \sigma_i \wedge u \upharpoonright B, C \in \tau_j)\} = \\ &= \bigcup_{i \in I, j \in J} \{u \upharpoonright A, C \mid u \in \mathcal{L}_{A,B,C} \wedge u \upharpoonright A, B \in \sigma_i \wedge u \upharpoonright B, C \in \tau_j\} = \\ &= \bigcup_{i \in I, j \in J} (\tau_j \circ \sigma_i)\end{aligned}$$

□

*Category.* We can form a category $\mathcal{H}$ whose objects are games and with morphisms from $A$ to $B$ the strategies on $A^{\perp} \circledast B$. With abuse of notation we write $\sigma$ for both the morphism $\sigma : A \to B$ and the strategy $\sigma : A^{\perp} \circledast B$. The identity morphism $id_A : A \to A$ is the well-known copycat strategy:

$$id_A = \{s \in \mathcal{L}_{A^{\perp} \circledast A} \mid s \upharpoonright A_1 = s \upharpoonright A_2\},$$

where the indices are used only to distinguish the left and the right copy of $A$. It is easy to check that $id_A$ is a well-defined prestrategy, that it is positional and deterministic, and that the identity equations

$$\sigma \circ id_A = \sigma = id_B \circ \sigma,$$

hold. There only remains the proof of associativity of composition, which we do next.

**Theorem 3.11** *Consider games $A$, $B$, $C$ and $D$, and strategies $\sigma : A \to B$ and $\tau : B \to C$ and $\rho : C \to D$:*

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

**Proof.** Let $s \in (\rho \circ \tau) \circ \sigma$. There is a witness $u \in \mathcal{L}_{A,B,D}$ for $s$. Analogously we find a witness $v \in \mathcal{L}_{B,C,D}$ for $u \upharpoonright B, D$. Working in an asynchronous world sometimes makes things easier (but only sometimes) so now we need no zipping or infinite zipping lemma, we can interleave $u$ and $v$ in no matter which way, we always end up with some $w \in \mathcal{L}_{A,B,C,D}$ which satisfies:

- $w \upharpoonright A, B = u \upharpoonright A, B \in \sigma$;
- $w \upharpoonright B, C = v \upharpoonright B, C \in \tau$;
- $w \upharpoonright C, D = v \upharpoonright C, D \in \rho$;
- $w \upharpoonright A, D = u \upharpoonright A, D = s$.

Then by definition we have $w \upharpoonright A, C \in \tau \circ \sigma$ and $s \in \rho \circ (\tau \circ \sigma)$. The opposite direction is proved in exactly the same way. □

Lifting $\circledast$ to morphisms allows one to derive the description of a circuit from the descriptions of its independent components. Given $\sigma : A \to B$ and $\sigma' : A' \to B'$, we define $\sigma \otimes \sigma' : A \otimes A' \to B \otimes B'$ as

$$\sigma \otimes \sigma' = \{s \in \mathcal{L}_{A,A',B,B'} \mid s \upharpoonright A, B \in \sigma \wedge s \upharpoonright A', B' \in \sigma'\}$$

The functoriality properties, $(\tau \circ \sigma) \otimes (\tau' \circ \sigma') = (\tau \otimes \tau') \circ (\sigma \otimes \sigma')$ and $id_A \otimes id_B = id_{A \otimes B}$, are easily verified. Suppose we want to compose strategies $\sigma : A \to B$ and $\tau : B \otimes B' \to C$. A possible way of doing this is by 'extending' $\sigma$ with the identity of $B'$. The composition becomes $\tau \circ (\sigma \otimes id_{B'})$. Another way is to restrict $\tau$ and substitute it with $\tau \circ (id_B \otimes SKIP_{B' \to I})$, where $I$ is the game with just no moves.

The category of handshake games and handshake strategies $\mathcal{H}$ is **-autonomous*, it is symmetric monoidal closed with respect to $\otimes$ and has an isomorphism from $A$ to $A^{\perp\perp}$. In particular the isomorphisms $A \to I \otimes A$ and $A \to A \otimes I$ are just copycat strategies and they follow from the isomorphisms between sets of moves, $M_A \to \emptyset + M_A$ and $M_A \to M_A + \emptyset$, respectively. The other isomorphisms required for symmetric monoidal closure are also copycat strategies derived from isomorphisms between sets of moves. The case of self-duality is even simpler, $A$ and $A^{\perp\perp}$ are the same game and the isomorphism between them is just the identity on $A$.

We can further note that $\mathcal{H}$ is more than *-autonomous, it is compact-closed, as you can easily check $A \otimes B = (A^{\perp} \otimes B^{\perp})^{\perp}$. Computationally this feature allows the modeling of loops. To see this, remember that arrows are just games of the form $A^{\perp} \otimes B$ (this is the essence of compact-closedness) and that the isomorphisms of symmetric monoidal closure allow to move freely subcomponents from one side to the other. Then the trick is easily accomplished by composing the desired subcomponents together, for example with a wire (the identity or, better, a strategy isomorphic to it).

## 4 Semantics

Tangram is a Hardware Design Language in which programs are compiled directly into circuits constructed from a set of standard handshake components. Van Berkel proposes a semantics for Core Tangram [11] [28] which he calls calculus of handshake processes and is a calculus derived from Hoare's CSP. Then he proves a compilation theorem which states that the circuit resulting from the compilation of a Tangram program is equivalent to the meaning of that program in the calculus of handshake processes. In an alternative approach, handshake circuits themselves can be used as a semantics. Handshake components can be seen as special combinators which are described as handshake strategies, thus morphisms in $\mathcal{H}$. For simplicity I use the same uniform compilation strategy as in [28], and to keep this short I will also skip the syntax of Tangram and the compilation function which can be found on the same paper. So there only remains to define the required components in terms of strategies. I already defined the strategies for the three most general components, $RUN$, $SKIP$ and $STOP$. Each one of them is actually a family of components,

[11] Where data are not considered.

parameterized by the particular structure it is implemented on, and similarly there is a family of strategies describing it. The other required components are all implemented on particular structures. The *constant* component is implemented on a passive output port. Semantically this corresponds to a game with an opponent request $r$, the values $v$ from a set $V$ as acknowledges and a single dependency class (the one containing all moves). Then the associated strategy is

$$CST_c = \{\varepsilon, rc, rcrc, \ldots\}$$

where $c \in V$. A bit more complex, consider a structure with two nonput ports, one passive ($A$) and one active ($B$). And here are our strategies on such a game:

$$\begin{aligned}
CON_{A,B} &= \{\varepsilon, r_A r_B, r_A r_B a_B a_A, \ldots\}\\
REP_{A,B} &= \{\varepsilon, r_A r_B, r_A r_B a_B r_B, \ldots\}\\
REPN_{A,B} &= \{\varepsilon, r_A r_B, \ldots, r_A r_B \ldots (n \text{ handshakes on } B) \ldots a_B a_A, \ldots\}
\end{aligned}$$

Where the indices on moves are used to distinguish the component over which these moves are played. As for the labels $r_A$ and $r_B$ are requests, $a_A$ and $a_B$ are acknowledges, $r_A$ and $a_B$ are opponent moves, $a_A$ and $r_B$ are player moves. The *connector* behaves like a wire. $REP$ is initially activated by a request on $A$ and then handshakes infinitely many times on $B$. $REPN$ does the same, except that after the $N$th time it stops handshaking on $B$ and acknowledges the initial request on $A$, waiting for a new one to come. Adding data to both acknowledges allows us to define two more strategies that we need. In the game we consider $a_A$ is replaced by the set $\{v_A \mid v \in \tau_A\}$ and $a_B$ by $\{v_B \mid v \in \tau_B\}$, where $\tau_A$ and $\tau_B$ are two sets of values. The rest of the structure is modified accordingly. So we have:

$$\begin{aligned}
ADAPT_{\tau_A,\tau_B} &= \{\varepsilon, r_A r_B\} \cup \{r_A r_B v_B v_A \mid v \in \tau_A\} \cup \ldots\\
UN_{\Box} &= \{\varepsilon, r_A r_B, r_A r_B v_B (\Box v)_A, \ldots\}
\end{aligned}$$

They both behave similarly to $CON$ with the addition of data in the acknowledges. When possible the *adapter* transmits the value received on $B$ through $A$, otherwise it is undefined. Assuming $\Box$ is a unary function from $\tau_B$ to $\tau_A$, $UN$ applies it to the value received from $B$ and sends the result on $A$. Back to the case of nonput ports we add an active port $C$. The strategies we consider are:

$$\begin{aligned}
SEQ_{A,B,C} &= \{\varepsilon, r_A r_B, r_A r_B a_B r_C, r_A r_B a_B r_C a_C a_A, \ldots\}\\
OR_{A,B,C} &= \{\varepsilon, r_A r_B, r_A r_C, r_A r_B a_B a_A, r_A r_C a_C a_A, \ldots\}\\
PAR_{A,B,C} &= \{\varepsilon, r_A r_B r_C, r_A r_C r_B, r_A r_B r_C a_B, r_A r_B a_B r_C, r_A r_C r_B a_B, r_A r_B r_C a_C,\\
&\qquad r_A r_C a_C r_B, r_A r_C r_B a_C, r_A r_B r_C a_B a_C a_A, r_A r_B r_C a_C a_B a_A, r_A r_B a_B r_C\\
&\qquad a_C a_A, r_A r_C r_B a_B a_C a_A, r_A r_C r_B a_C a_B a_A, r_A r_C a_C r_B a_B a_A, \ldots\}
\end{aligned}$$

They are all activated by the request on $A$. Then the *sequencer* starts two handshakes in sequence on the active ports, the $PAR$ starts them in parallel and the $OR$ does either of the two (nondeterministically). Afterwards they all acknowledge to the first request. $SEQ$ denotes the standard directive of sequential composition of commands and should not look new to the game semantics community, basically any model of (a variant of) *Ideal Algol* will do (to my knowledge, the first of these models is described in [3]). Similarly $PAR$ denotes the standard directive for parallel execution of commands and has also been modeled with games [19][11]. Basically any variant of $PAR$ on a game with values in place of the plain acknowledges will give semantics to some binary operator:

$$BIN_{\Box} = \{\varepsilon, r_A r_B r_C, \ldots, r_A r_B r_C v_B w_C (v \Box w)_A, \ldots\}$$

Variants of the sequencer are also possible. For example if we add values (of the same type) in the acknowledges of $B$ and in the requests of $C$, we can define the *transferrer*:

$$TRF_{A,B,C} = \{\varepsilon, r_A r_B, r_A r_B v_B v_C, r_A r_B v_B v_C a_C a_A, \ldots\}$$

After activation, the transferrer asks for a value from $B$ which is then sent over to $C$. Another variant of the sequencer (but on a different game) allows for representation of the branching construct:

$$IF_{A,B,C} = \{\varepsilon, r_A r_B, r_A r_B tt_B r_C, r_A r_B tt_B r_C a_C a_A, \ldots\}$$

The only data are in the acknowledges of $B$ and are booleans. Straight to the crucial point: if the acknowledge on $B$ contains $tt$ then $IF$ starts a handshake on $C$, otherwise the behavior is undefined (it may decide to stop and wait or to continue in any possible way). Still on the same game we can define a strategy which allows looping:

$$DO_{A,B,C} = \{\varepsilon, r_A r_B, r_A r_B tt_B r_C, r_A r_B ff_B a_A, r_A r_B tt_B r_C a_C r_B, \ldots\}$$

We now move to structures with two passive ports. The first game we consider has as set of moves the disjoint sum of sets $M_A = \{r_A\} \cup \{v_A \mid v \in V\}$ and $M_B = \{v_B \mid v \in V\} \cup \{a_B\}$, where $V$ is a given set of values. The dependency relation is as expected and the labelling assigns $(-, R)$ to $r_A$ and all $v_B$'s, and $(+, A)$ to $a_B$ and all $v_A$'s. The *variable* is defined by the strategy

$$VAR_{A,B} = \{\epsilon, r_A m_A, \ldots, \ldots n_B a_B r_A n_A r_A n_A \ldots, \ldots\}$$

To the first read request it may respond in any way (undefined behavior) but once a value is written, that value will be returned at any following read (until a new value is written over). The next game represents a structure with two passive ($A$

and $B$) and one active ($C$) ports, all nonput. Two strategies on this game:

$$MIX_{A,B,C} = \{\varepsilon, r_A r_C, r_A r_C a_C a_A, \ldots, r_B r_C, r_B r_C a_C a_B, \ldots\}$$
$$JOIN_{A,B,C} = \{\varepsilon, r_A, r_B, r_A r_B r_C, r_A r_B r_C a_C a_A a_B, \ldots\}$$

The *mixer* mixes two communication flows into a single one. Whenever it receives a request on either passive port it sends a request on the active one, when this is acknowledged, it acknowledges the request it received. The *join* joins two concurrent threads. It initially waits for both requests from $A$ and $B$ to arrive, then it sends a request on $C$. When this is acknowledged, it sends acknowledges on $A$ and on $B$ and returns to the initial state. The last component is needed for Tangram's guarded-commands and has by far the most complex structure, two passive and four active ports, with boolean values in the acknowledgements of one passive port (an output port) and of two active ports (two input ports). The remaining three are nonput ports. The associated game contains the set of moves $\{r_B, tt_B, ff_B, r_C, a_C, r_{BL}, tt_{BL}, ff_{BL}, r_{CL}, a_{CL}, r_{BR}, tt_{BR}, ff_{BR}, r_{CR}, a_{CR}\}$. The dependency relation is as expected and the labels are all request for the $r_*$ moves and acknowledge for the others; moreover $r_B$ and $r_C$ are opponent moves and the corresponding acknowledges are player moves, all the other requests are player moves and the corresponding acknowledges are opponent moves. The $BAR$ component initially waits for $r_B$, then sends in parallel $r_{BL}$ and $r_{BR}$. It will receive two boolean values in the acknowledges, then it does the logical or of these two and sends the result as acknowledge to $r_B$. At that point, once $r_C$ is arrived, it starts behaving according to the boolean values previously received in the acknowledges to $r_{BL}$ and $r_{BR}$. If only one of them was true it sends the paired request [12] ; if both of them were it chooses nondeterministically which request to send; finally if both values were false then the behavior is left unspecified. Here is the strategy:

$$BAR_{B,C} = \{\epsilon, r_B r_{BL} r_{BR}, \ldots, r_B r_{BL} r_{BR} b_{BL} b'_{BR} (b \vee b')_B, \ldots, r_B r_{BL} r_{BR} b_{BL} tt_{BR}$$
$$tt_B r_C r_{CR} a_{CR} a_C, \ldots\}$$

For clarity reasons, in the last four examples I only wrote the plays that help understand the internal logic of the components, you may think that the other plays are hidden behind the dots. In fact, properties such as receptivity and reorder-closedness imply that several other 'sequentializations' of the global concurrent behavior may take place [13] . Still, all these components have cyclic behaviors, they sooner or later return to previously seen states, then a fully satisfactory description with all possible plays that may occur in a cycle (like the one I gave for $PAR$) is possible.

[12] Where pairing corresponds to intuition: ports $BL$ and $CL$ form one pair, and ports $BR$ and $CR$ form the other.

[13] For example, in the case of $BAR$, $r_c$ could arrive at any time before the acknowledge to $r_B$, even before $r_B$ itself.

## 5 Conclusions and Further Work

So, I built a model for handshake circuits and I showed how it could be put into use as a semantics of Tangram, bridging two apparently distant worlds: games and circuits. In the process, I dealt with the problems arising in the modeling of circuits' composition and I ended with a finer characterization of handshake strategies. In particular, the new property of positionality seems very interesting and worth exploiting. But the model contains also a deeper structural analysis, with the construction of a compact-closed category $\mathcal{H}$. This provides us with tools like composition of morphisms and the bifunctor $\otimes$ which turn out useful: once I had defined such a model the following step was almost immediate. I considered the language Tangram and took advantage of the compilation function proposed by Van Berkel [28], from Tangram programs to handshake circuits. The obtained circuits were all constructed from standard handshake components connected together. Then these components can be seen as combinators and represented as handshake strategies in $\mathcal{H}$.

The compact-closedness of $\mathcal{H}$ allows for the modeling of internal loops, which is already a good thing; however a finer characterization of this category can be given. One can take advantage from models of the *Geometry Of Interaction* [12][5][9] and make structural comparison. The aim is to provide a general framework which may contain larger classes of asynchronous circuits [14]: in this sense, the categorical way is an especially feasible one. As starting point we may take the family of process algebras developed by Mark Josephs [17][16][18] on one hand of the bridge, and the implementations of the geometry of interaction with circuits [20] on the other.

This continous "river-crossing", shifting the viewpoint from circuits to games/GOI and back, might return positive feedbacks to both ends. By the formal banks of game semantics, the community could benefit from an implementation of their formalism with real physical devices. Moreover, the use of handshake components as combinators for the semantics of a language looks promising: one could think of using the same combinators in the semantics of languages which are radically different from Tangram. At the other side of the river, some modest contributions already got ashore along with this work, as I have noted above. More feedback can come from parallelisms with games for the verification of properties [24][10][25]. An orthogonal direction is in the abstraction of the formalism. I already pointed out that the standard handshake components all follow a cyclic life, then this suggests more abstract representations, ways that could allow to treat composition more mathematically. The aim is to find alternatives for simplifying reasoning while keeping the substance unchanged.

As far as semantics is concerned, the current work can be seen as an extension of Van Berkel's, who only gives a semantics to the fragment of Tangram without data. Surely there is more work to do: the focus of this paper was on the construction of a model for handshake circuits, along with full details on the description of the solution to the pre-existing problem of associativity of composition, while the

[14] Two of these classes are *speed-independent* and *delay-insensitive* circuits.

semantical aspect still needs to be regarded more systematically. First of all, in my opinion, it is not completely clear in which direction to look at the denotation function. One can for example consider circuits and model them with a calculus which in a way mimics the language, as in [28]; but one can as well use circuits directly as a semantics for the language, as a description of what a program does. Since the compilation of a program yields a circuit, two programs should be considered equivalent just when they yield two observationally equivalent circuits[15]. Then a property like full abstraction for a model which describes so closely the external behavior of circuits (for a model like mine) becomes trivial. But of course other interpretations of program equivalence in Tangram may be possible.

## References

[1] S. Abramsky. Interaction categories (extended abstract). In G. L. Burn, S. J. Gay, and M. D. Ryan, editors, *Theory and Formal Methods 1993: Proceedings of the First Imperial College Department of Computing Workshop on Theory and Formal Methods*, pages 57–70. Springer-Verlag Workshops in Computer Science, 1993.

[2] 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-Verlag, Berlin, 1996.

[3] S. Abramsky and G. McCusker. Linearity, sharing and state: a fully abstract game semantics for idealized algol. In P. O'Hearn and R.D. Tennent, editors, *Algol-like Languages*, pages 317–348. Birkhauser, 1997.

[4] S. Abramsky and P.-A. Melliès. Concurrent games and full completeness. In *Logic in Computer Science 99*, pages 431–442, Trento, July 1999. IEEE Computer Society Press.

[5] Samson Abramsky and Radha Jagadeesan. New foundations for the geometry of interaction. *Information and Computation*, 111(1):53–119, 1994. Conference version appeared in LiCS '92.

[6] A. Bardsley. Balsa: an Asynchronous Circuit Synthesis System. master's thesis. Department of Computer Science, University of Manchester, 1998.

[7] I. Benko. Theory of handshake circuits. highlights, 1994.

[8] Andreas Blass. A game semantics for linear logic. *Annals of Pure and Applied Logic*, 56(1-3):183–220, 1992.

[9] V. Danos and L. Regnier. Proof-nets and the hilbert space. In *Proceedings of the workshop on Advances in linear logic*, pages 307–328, New York, NY, USA, 1995. Cambridge University Press.

[10] D.R. Ghica and A.S. Murawski. Compositional model extraction for higher-order concurrent programs. In *TACAS'06, Lecture Notes in Computer Science 3920*, pages 303–317. Springer-Verlag, 2006.

[11] D.R. Ghica, A.S. Murawski, and C.-H. L. Ong. Syntactic control of concurrency. In *ICALP'04, Lecture Notes in Computer Science 3142*, pages 683–694. Springer-Verlag, 2004.

[12] J.-Y. Girard. Geometry of interaction I: interpretation of system F. In Ferro, Bonotto, Valentini, and Zanardo, editors, *Logic Colloquium '88*, pages 221–260, Amsterdam, 1989. North-Holland.

[13] Russell Harmer and Guy McCusker. A fully abstract game semantics for finite nondeterminism. In *Logic in Computer Science*, pages 422–430, 1999.

[14] Russell Harmer and Guy McCusker. A game semantics for countable nondeterminism. Unpublished, 2002.

[15] M. Hyland and A. Schalk. Games on graphs and sequentially realizable functional. In *IEEE Computer Society Press*, pages 257–264, Kopenhavn, July 2002.

[16] Mark B. Josephs. Receptive process theory. *Acta Inf.*, 29(1):17–31, 1992.

[15] Note that this view of program equivalence is consistent with the algebraic treatment of handshake processes given in [18].

[17] Mark B. Josephs and Jan Tijmen Udding. An algebra for delay-insensitive circuits. In *CAV '90: Proceedings of the 2nd International Workshop on Computer Aided Verification*, pages 343–352, London, UK, 1991. Springer-Verlag.

[18] 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.

[19] J. Laird. A game semantics of idealized csp. *Electronic notes in Theoretical Computer Science*, 45, 2001. Seventeenth Conference on Mathematical Foundations of Programming Semantics, MFPS'01.

[20] Ian Mackie. The geometry of interaction machine. In *POPL '95: Proceedings of the 22nd ACM SIGPLAN-SIGACT symposium on Principles Of Programming Languages*, pages 198–208, New York, NY, USA, 1995. ACM Press.

[21] A. Mazurkiewicz. Trace theory. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Advances in Petri Nets*, pages 279–324. Springer, 1986.

[22] P.-A. Melliès. Asynchronous games 2: the true concurrency of innocence. In P. Gardner and N. Yoshida, editors, *CONCUR'04, Lecture Notes in Computer Science*. Springer Verlag, September 2004.

[23] Samuel Mimram and Paul-André Melliès. Asynchronous games 5: Non-alternating innocence. Work in progress, 2006.

[24] Y. Moschovakis. A game-theoretic modeling of concurrency. In *fourth annual symposium on Logic In Computer Science*, pages 154–163. IEEE Computer Society Press, 1989.

[25] C.-H. L. Ong. Some results on a game-semantic approach to the verification of infinite structures. To appear in the proceedings of CSL 2006.

[26] D. Park. *The "fairness" problem and nondeterministic computing networks.* Foundations of Computer Science IV. Matematisch Centrum, Amsterdam, 1983.

[27] A. W. Roscoe. *Unbounded nondeterminism in CSP*, volume 3 of *Journal of Logic and Computation.* April 1993.

[28] K. Van Berkel. *Handshake circuits: an Asynchronous architecture for VLSI design*, volume 5 of *Cambridge International Series on Parallel Computation.* Cambridge University Press, 1993.

## 6 Appendix: Associativity in the Calculus of Handshake Processes

Associativity of composition is often a "tough beast", especially when dealing with processes which may present degrees of parallelism and/or of nondeterminism [8][13]. In this appendix I claim, and show with a counter-example, that the problem has been underestimated in Van Berkel's Calculus of Handshake Processes. I will reformulate the example in terms of handshake games and handshake prestrategies (handshake processes can be redefined equivalently as handshake prestrategies with no infinite plays). Even though infinite plays are not explicitly there, the case of infinite chattering between two prestrategies does not disappear and composition requires a special treatment in this particular case: here is where the problem lies. In particular, composition for finite plays is as defined in section 3. Moreover, consider those infinite interleavings of the two strategies, $\sigma$ and $\tau$, whose finite prefixes are all in the prefix-closures of $\sigma$ and $\tau$ (under the proper restriction of course). We complete the composed strategy by adding the plays obtained from exactly these interleavings after hiding internal threads.

Here follows the counter-example. Let $Act = \langle \{r_A, a_A\}, D_A, \lambda_A \rangle$ be the game associated to a nonput active port ($r_A$ and $a_A$ are dependent with each other, $\lambda_A(r_A) = (+, R)$, $\lambda_A(a_A) = (-, A)$). Let $Pas = \langle \{r_P, a_P\}, D_P, \lambda_P \rangle$ be the game associated to a nonput passive port $(\ldots)$. Finally, let $Sing = \langle \{r_S\}, (r_S, r_S), r_S \mapsto (+, R) \rangle$ be the game containing a single player request. Now consider for a moment the handshake process [16] $RUN_{Act \to Sing} : Act \to Sing$, which might also be regarded as $RUN_{Pas \otimes Sing}$, or as $RUN_{Pas} \otimes RUN_{Sing}$. Then it is actually composed of two prestrategies acting independently and concurrently: the first one replies to each request coming from the opponent on $Pas$, the second one does the only thing it can possibly do, it sends a request on $Sing$. The prestrategy is quiescent just after the request on the right side has been sent while on the left side there is no request which is still pending (unanswered). The other prestrategies are $RUN_{I \to Act} : I \to Act$ and $STOP_{Sing^{\perp}} \otimes RUN_{Sing} : Sing \to Sing$. In particular this last prestrategy becomes quiescent once it has sent the request on the right, both if the request on the left is already arrived or not.

We start with $RUN_{I \to Act}$ and $RUN_{Act \to Sing}$, composing them Van Berkel's style. As a result we get a nondeterministic prestrategy on $I \to Sing$ which might stay quiescent or send a request, let's call it $\tau$. If we compose $\tau$ with $STOP_{Sing^{\perp}} \otimes RUN_{Sing}$ we get $RUN_{I \to Sing}$ as a result. It can be easily seen that doing the compositions in the other order, the obtained result is $\tau$: two different results then. Note also that using this style of composition, two deterministic processes may compose into a nondeterministic one. In $\mathcal{H}$, for $RUN_{Act \to Sing}$ to be deterministic it must contain all and only the infinite plays where the request on $Sing$ appears at some point. A nondeterministic variant of this strategy can be possible as well: in any case the final composition yields the unique play $RUN_{I \to Sing}$, as expected.

Note that this counterexample has nothing to do with nondeterminism apriori, we are trying to observe the interaction of deterministic strategies, even if we do it inside a model specific for nondeterminism. Then it is yet a new counter-example, different from Roscoe's one. Interestingly, they both bring to the same conclusion: that in a model of unbounded nondeterminism infinite plays (traces, dialogues, ...) must be listed explicitly.

[16] I call it handshake process in order to stress the fact that it does not contain infinite plays.