Game Semantics for Higher-Order Concurrency

J. Laird*

Department of Informatics, University of Sussex, UK

Abstract. We describe a denotational (game) semantics for a call-by-value functional language with multiple threads of control, which may communicate values of general type on locally declared channels.
This develops previous work which interpreted freshly generated names in a category of games acted upon by the group of natural number automorphisms, by showing how names may be associated with "dependent arenas" in which interaction between strategies, corresponding to asynchronous communication on named channels, may occur.
We describe a model of the call-by-value λ-calculus (a closed Freyd category) based on these arenas, and use this as the basis for interpreting our language. We prove that the semantics is fully abstract with respect to may-testing using a correspondence between channel and function types based on the "triggering" representation of procedure-passing in terms of name-passing.

1 Introduction

Higher-order concurrency — the capacity to generate multiple threads of control and pass higher-order functions and processes as values between them — is a powerful and subtle programming paradigm. Languages and calculi with these features, such as *Concurrent ML* and the higher-order π-calculus, have been extensively studied using operational methods, but the combination of dynamic name creation and higher-order value-passing has presented a long standing challenge for denotational semantics. In this paper, we shall develop a denotational model of a call-by-value functional language with higher-order concurrency, including dynamically generated channel names, and prove that it is fully abstract with respect to may-testing.

Our model is based on *game semantics*. This has proved successful in giving precise (fully abstract) models of higher-order programming languages with many different features, including concurrency [3, 12]. Our semantics opens the door to a range of (largely theoretical) applications: by formalizing some of the categorical and algebraic structures required to capture higher-order concurrency, it can contribute to the development of general theories, whilst also potentially being the basis for more specific forms of program analysis, already developed for various games models, such as control and information flow analysis [14], and model-checking of properties such as program equivalence [2, **?**].

* E-mail: jiml@sussex.ac.uk. Research supported by UK EPSRC grant GR/S72181.

1.1 Related Work

Our semantics is given for a language with syntax based on Reppy's Concurrent ML [17]: it may also be viewed as a programming language variant of the higher-order π-calculus [18]. Both languages (or fragments thereof) have been investigated using operational techniques [18, 1, 8, 9], giving, for example, labelled transition systems which are closely related to game semantics. Indeed, our semantics may be viewed as a trace semantics, defined compositionally. (See [12] for an explicit comparison between game and trace semantics of the π-calculus.)

Our model uses (and adapts) a variety of notions from game semantics, including the representation of call-by-value types introduced by Honda and Yoshida [5]. In common with earlier models of shared-variable concurrency [3] and the π-calculus [12], we represent terms up to asynchronous, may-testing observation as sets of justified sequences (traces) closed under a preorder. Particularly significant for the current work is the development of a category of "ν-arenas and ν-strategies" acted upon by the group of natural number automorphisms [11], with which the manipulation and generation of *names* can be interpreted (in the current case, channel names). A certain degree of parametricity is implicit in the representation of values at a range of types as names, and there are parallels in this respect between our games and the game semantics of polymorphism described by Hughes [6].

1.2 Contribution of this Paper

The main technical contribution of this paper is to show how the game semantics of freshly generated names developed in [11] may be used to model the passing of values of all types on named, typed channels. The construction which makes this possible is a notion of "tree arena" which has ν-arenas as its nodes, each of which has as its children a "dependent arena" for each name which may be mentioned by each move. Using a name allows interaction to take place in its dependent arena, corresponding to message passing on the associated channel. We define a category of games in which we define "parallel composition plus hiding" of strategies to allow interaction both at top level, and within dependent arenas with names which have been made public. We then define the structure of a categorical model of the call-by-value λ-calculus (a premonoidal closed category), and interpret the key operations of our language (spawning of threads, generation of channels, sending and receiving of messages) as simple strategies. We show that our interpretation is sound and adequate with respect to may-testing.

We then prove that the bounded elements of our semantics are definable as terms of $\mathcal{L}$, and hence that it is fully abstract with respect to may-testing. The key to the proof of definability is the observation that justification pointers may be encoded as names using a series of definable retractions. This is a semantic counterpart of Sangiorgi's *triggering* translation from higher-order processes into the π-calculus [18]. We may then give a simple proof that "pointer-free" sequences are definable as π-calculus-like terms.

2 A Language with Higher-Order Concurrency

The programming language $\mathcal{L}$ which we shall interpret contains several of the key features of CML [17]: new thread generation, new channel declaration (it omits thread identifiers, which are readily expressible using channel names, and event types). Thus it is similar both to μCML [1] and $\mu\nu$CML [9] (although, unlike these languages, communication is *asynchronous*). The types of $\mathcal{L}$ are given by the grammar:
$S, T := B \mid S * T \mid S \Rightarrow T \mid \mathtt{chan}[T]$ where B is a set of basic types including at least `unit`, `bool`, and an empty type `0`. The syntax and typing judgements of $\mathcal{L}$ are those of the typed λ-calculus extended with the following constants:

Pairing/Projection $\mathtt{pair} : S \Rightarrow T \Rightarrow S * T$, $\mathtt{fst} : S * T \Rightarrow S$, $\mathtt{snd} : S * T \Rightarrow T$
Atomic Values $() : \mathtt{unit}$ and $\mathtt{tt}, \mathtt{ff} : \mathtt{bool}$,
Conditional $\mathtt{If} : \mathtt{bool} \Rightarrow (T * T) \Rightarrow T$
Equality Testing $\mathtt{eq} : \mathtt{chan}[T] * \mathtt{chan}[T] \Rightarrow \mathtt{bool}$,
Channel Declaration $\mathtt{new}_T : \mathtt{unit} \Rightarrow \mathtt{chan}[T]$,
Thread Creation $\mathtt{spawn} : (\mathtt{unit} \Rightarrow \mathtt{0}) \Rightarrow \mathtt{unit}$,
Message Passing $\mathtt{send} : \mathtt{chan}[T] * T \Rightarrow \mathtt{0}$ and $\mathtt{recv} : \mathtt{chan}[T] \Rightarrow T$

We write (M, N) for $(\mathtt{pair}\, M)\, N$, $\nu x.M$ for $(\lambda x.M)\,(\mathtt{new}\,())$, $\mathtt{nil}$ for $\nu c.\mathtt{recv}\, c$, $M = N$ for $(\mathtt{eq}\,(M, N)$ and $M|N$ for $(\mathtt{spawn}\,(\lambda x.M)); N$. We may express recursively defined functions as: $\mu f.\lambda x.M =_{df}$
$\nu c.\mathtt{let}\ f = (\lambda x.\mathtt{let}\ g = \mathtt{recv}\, c\ \mathtt{in}\ (\mathtt{send}\,(c, g) | g\, x))\ \mathtt{in}\ (\mathtt{send}\,(c, \lambda x.M) | f)$.
In particular, we define the replicated send: $!\mathtt{send}\, M =_{df} \mu f.\lambda x.(\mathtt{send}\,(c, M) | f\,())$.

A *configuration* of consists of a multiset $\mathcal{T}$ of programs or *threads* $M_1, \ldots, M_n$ (of which at most one has non-empty type), and a set $\mathcal{N}$ of typed channel names such that $\mathcal{N} \vdash M_i$ for $1 \leq i \leq n$. The *values* of $\mathcal{L}$ are given by the grammar:
$U, V ::= v \mid n \mid C \mid \mathtt{pair}\, U \mid (\mathtt{pair}\, U)\, V \mid \lambda x.M$
(where v ranges over base type values, n over channel names and C over constants). The *evaluation contexts* are: $E[\_] ::= [\_] \mid E[\_]\, M \mid V\, E[\_]$.
The "small step" evaluation rules for evaluating configurations are as follows:

$$
\begin{array}{lcl}
\mathcal{T}, E[(\lambda x.M)\, V], \mathcal{N} & \longrightarrow & \mathcal{T}, E[M[V/x]], \mathcal{N} \\
\mathcal{T}, E[\mathtt{fst}(U, V)], \mathcal{N} & \longrightarrow & \mathcal{T}, E[U], \mathcal{N} \\
\mathcal{T}, E[\mathtt{snd}(U, V)], \mathcal{N} & \longrightarrow & \mathcal{T}, E[V], \mathcal{N} \\
\mathcal{T}, E[\mathtt{eq}(a, a)], \mathcal{N} & \longrightarrow & \mathcal{T}, E[\mathtt{tt}], \mathcal{N} \\
\mathcal{T}, E[\mathtt{eq}(a, b)], \mathcal{N} & \longrightarrow & \mathcal{T}, E[\mathtt{ff}], \mathcal{N},\ \text{if } a \neq b \\
\mathcal{T}, E[\mathtt{If}\ \mathtt{tt}], \mathcal{N} & \longrightarrow & \mathcal{T}, E[\mathtt{fst}], \mathcal{N} \\
\mathcal{T}, E[\mathtt{If}\ \mathtt{ff})], \mathcal{N} & \longrightarrow & \mathcal{T}, E[\mathtt{snd}], \mathcal{N} \\
\mathcal{T}, E_1[\mathtt{send}(a, V)], E_2[\mathtt{recv}(a)], \mathcal{N} & \longrightarrow & \mathcal{T}, E_2[V], \mathcal{N} \\
\mathcal{T}, E[\mathtt{spawn}(V)], \mathcal{N} & \longrightarrow & \mathcal{T}, V\,(), E[()], \mathcal{N} \\
\mathcal{T}, E[\mathtt{new}_T\,()], \mathcal{N} & \longrightarrow & \mathcal{T}, E[a], \mathcal{N} \cup \{(a, \mathtt{chan}[T])\} \quad a \notin \mathcal{N}
\end{array}
$$

We write $M \Downarrow$ (M *may* converge) if $M, 0 \twoheadrightarrow (\mathcal{T}, V), \mathcal{N}$ for some $\mathcal{T}, V, \mathcal{N}$. Thus we define observational approximation and equivalence with respect to may-testing:
$M \lesssim N$ if $C[M] \Downarrow$ implies $C[N] \Downarrow$ for all compatible closing contexts $C[\_] : \mathtt{unit}$.
$M \simeq N$ if $M \lesssim N$ and $N \lesssim M$.

3 Game Semantics

Our notion of game is based on the dialogue games of Hyland and Ong [7], developed in e.g. [15, 5] and extended in [11] with structure for manipulating a countable set of names, in the form of an action of the automorphism group of the natural numbers. A significant departure from these games for sequential languages arises because in the concurrent setting there are moves which may be played by either participant in a dialogue. So polarity (Player/Opponent labelling) is not intrinsic to arenas but to interactions.

An (underlying) arena A is a tuple $(M_A, \lambda_A, \vdash_A)$ consisting of a set of moves M_A, a question/answer labelling $\lambda_A : M_A \to \{Q, A\}$ and an *enabling relation* $\vdash_A \subseteq M_A \times M_A$ such that no answer enables an answer. We write M_A^I for the subset of M_A of consisting of moves with no enabling move (the *initial* moves), and say that an arena is *A-rooted* if all such moves are answers.

A justified sequence over the arena A is a sequence of moves of A together with a "justification pointer" from each non-initial move to some enabling move.

Definition 1. *A (partial or total) polarization for a justified sequence s is a (partial or total) labelling of the occurrences of moves in s as belonging to Player and Opponent (concretely, a function λ^{OP} from the non-empty prefixes of s to $\{P, O\}$) such that the justifier of any Player move is an Opponent move and vice-versa. A polarized sequence t is a justified sequence with a total polarization. We write $t^\perp$ for the polarized sequence in which the labelling is reversed.*

We now recall the notion of ν-arena introduced in [11]. Let G be the topological group of automorphisms on $\mathbb{N}$ with the product topology on $\mathbb{N}^\mathbb{N}$. An action of G on a set A is continuous (with respect to the discrete topology on A) iff the stabiliser of any $a \in A$ is open in G and thus equal to the stabiliser of a finite subset $\nu(a) \subseteq \mathbb{N}$, the *support* of a.

Definition 2. *A ν-arena (A, π) is an underlying arena A together with a continuous action of G on M_A $(\cdot)$ such that $\lambda_A(\pi \cdot m) = \lambda_A(m)$ and $m \vdash n$ iff $\pi \cdot m \vdash \pi \cdot n$, which therefore extends pointwise to a continuous action on justified sequences of A. We write $\sim$ for the equivalence relation determined by this action — i.e. $s \sim t$ if $\exists \pi \in G.\pi \cdot s = t$.*

A key example is the ν-arena of names N, in which the set of moves is the set of natural numbers (all of which are initial moves), with the canonical action of G upon $\mathbb{N}$ — i.e. $M_N = M_N^I = \mathbb{N}$, $\lambda(i) = A$ for all i, and $\pi \cdot i = \pi(i)$.

For each polarized sequence s, we define the sets $P_\nu, O_\nu \subseteq_{fin} \mathbb{N}$ of new names introduced by Player and Opponent in s. $P_\nu(\varepsilon) = \varnothing$ and:

- $P_\nu(sa) = P_\nu(s) \cup (\nu(sa) - \nu(s))$ if a is Player move,
- $P_\nu(sa) = P_\nu(s)$ otherwise.
- $O_\nu(s) = \nu(s) - P_\nu(s)$.

In order to use names to represent channel types, we introduce a notion of "tree arena", in which each node is an arena, from which there are branches for each name occurring in the support of each move. Essentially, playing a move which mentions a name with a given "dependent arena" allows both Player and Opponent to commence play in that arena, corresponding to sending (if Player starts) or receiving (if Opponent starts) a value on the associated channel.

Definition 3. *A (finite-depth) tree arena is a ν-arena A, together with an indexed set $\{\alpha(m)_i \mid i \in \nu(m)\}$ of tree arenas for each move $m \in M_A$, such that:*

- *G-Invariance: for any $\pi \in G$, $m \in M_A$ and $i \in \nu(m)$, $\alpha(m)_i = \alpha(\pi \cdot m)_{\pi(i)}$.*
- *Finite Depth: there is no infinite chain of arenas $A \ll A_1 \ll A_2 \ll \ldots$, where $B \ll C$ if there exists $m \in M_B$ and $i \in \nu(m)$ such that $\alpha(m)_i = C$.*

Thus, for example, for any tree arena A, we may form the tree arena $Ch(A)$ which has as its root node the arena N, and as its children, copies of the arena A — i.e. $ch(A) = (N, \{\{\alpha(i)_i\} \mid i \in \mathbb{N}\})$, where $\alpha(i)_i = A$ for all i.

We refer to the set of nodes of a tree arena as its *dependent arenas*. Formally, this is defined by induction on tree depth as follows:
$|A| = \{\alpha(m)_i \mid m \in M_A \wedge i \in \nu(m)\} \cup \bigcup\{|\alpha(m)_i| \mid m \in M_A \wedge i \in \nu(m)\}$.

We obtain a tree arena $\widehat{A}$ — the *expansion of A* — by explicitly adding $\mathbb{N}$-indexed copies of the dependent arenas of A to the top node. More precisely:

- $M_{\widehat{A}} = M_A + \{\langle i, A, m\rangle \in \mathbb{N} \times |A| \times \bigcup\{M_B \mid B \in |A|\} \mid m \in M_B \wedge i \notin \nu(m)\}$,
- $m \vdash_{\widehat{A}} \mathrm{in}_{\mathrm{l}}(n)$ if $m = \mathrm{in}_{\mathrm{l}}(m')$ and $m' \vdash_A n$.
  $m \vdash_{\widehat{A}} \mathrm{in}_{\mathrm{r}}(\langle i, B, n\rangle)$, if $m = \mathrm{in}_{\mathrm{r}}(\langle i, B, m'\rangle)$, where $m' \vdash_B n$,
- $\lambda_{\widehat{A}}(\mathrm{in}_{\mathrm{l}}(m)) = \lambda_A(m)$ and $\lambda_{\widehat{A}}(\mathrm{in}_{\mathrm{r}}(\langle i, B, m\rangle)) = \lambda_B(m)$,
- $\pi \cdot \mathrm{in}_{\mathrm{l}}(m) = \mathrm{in}_{\mathrm{l}}(\pi \cdot_A m)$ and $\pi \cdot \mathrm{in}_{\mathrm{r}}(\langle i, B, m\rangle) = \mathrm{in}_{\mathrm{r}}(\pi(i), \pi \cdot_B m)$
- $\alpha^{\widehat{A}}(\mathrm{in}_{\mathrm{l}}(m))_i = \alpha^A(m)_i$
  $\alpha^{\widehat{A}}(\mathrm{in}_{\mathrm{r}}(\langle i, B, m\rangle))_i = B$, $\alpha^{\widehat{A}}(\mathrm{in}_{\mathrm{r}}(\langle i, B, m\rangle))_j = \alpha^B(m)_j$ if $j \neq i$.

Names of the form $\langle i, m\rangle$ are called *dependent moves.* The name of $\langle i, m\rangle$ is i.

Definition 4. *A* legal sequence *on A is a justified sequence s on $\widehat{A}$, satisfying:*

Uniformity *Every occurrence of a name refers to the same arena: if $ta, t'a' \sqsubseteq s$ and $i \in \nu(a) \cap \nu(a')$ then $\alpha(a)_i = \alpha(a')_i$.*
Dependency *The name of any dependent move has already occurred in s: if $t\langle i, a\rangle \sqsubseteq s$ then $i \in \nu(t)$.*
Well-openedness *s contains at most one initial and non-dependent move.*
Well-answering *Every question in s justifies at most one answer.*

A negative *sequence is a legal sequence starting with an Opponent move. We write L_A for the legal sequences over A, and L_A^- for the negative sequences.*

To represent the behaviour of strategies "up to asynchronous observation" requires saturation under a preorder $\preceq$ on polarized sequences as in e.g. [10]. This is defined to be the least preorder on polarized sequences such that:

- If $\lambda(a) = O$ and $P_\nu(sat) = P_\nu(st)$ then $sabt \preceq sbat$ and if $\lambda^{OP}(a) = P$ and $O_\nu(sat) = O_\nu(st)$ then $sbat \preceq sabt$.

– If $\lambda(a) = O$ and $P_\nu(sat) = P_\nu(st)$ then $sat \preceq st$, and if $\lambda(a) = P$ and $O_\nu(sat) = O_\nu(st)$, then $t \preceq sat$.

Definition 5. *Let A be a tree arena. A strategy $\sigma : A$ is a non-empty set of negative sequences over A which is prefix closed, $\sim$-closed and $\preceq$-closed. We write* $\ker(\sigma)$ *for the set of $\preceq$-minimal sequences of σ — i.e. $\{s \in \sigma \mid \forall t \in \sigma . s \preceq t \Longrightarrow t \preceq s\}$.*

4 Denotational Semantics

We will now construct a *premonoidal* closed category [16] of tree arenas and strategies in which to model the call-by-value λ-calculus. This follows the constructions of Honda and Yoshida [5] or variants described by in [13], and (for ν-arenas) [11]. In each case the group action and dependent arena structure on compound arenas is defined pointwise. Play in the function-space arena $A_1 \rightarrow A_2$ starts on the left (by labelling the initial moves of A_1 as questions which enable the initial moves of A_2).

Definition 6. *Given tree arenas A_1, A_2 we define the (Q-rooted) call-by-value function-space tree-arena $A_1 \rightarrow A_2$ as follows:*

– $M_{A_1 \rightarrow A_2} = M_{A_1} + M_{A_2}$,
– $\lambda_{A_1 \rightarrow A_2}(\mathrm{in}_i(m)) = Q$, *if* $i = 1$ *and* $m \in M^I_{A_2}$,
  $\lambda_{A_1 \rightarrow A_2}(\mathrm{in}_i(m)) = \lambda_{A_i}(m)$, *otherwise,*
– $m \vdash_{A_1 \rightarrow A_2} \mathrm{in}_i(n)$ *if* $m = \mathrm{in}_i(m')$ *and* $m' \vdash_{A_i} n$ *or* $i = 2$, $n \in M^I_{A_2}$ *and* $m = \mathrm{in}_1(m')$, *where* $m' \in M^I_{A_1}$.
– $\pi \cdot \mathrm{in}_i(m) = \mathrm{in}_i(\pi \cdot m)$.
– $\alpha(\mathrm{in}_i(m))_j = \alpha^{A_i}(m)_j$.

For example, for each arena A, we have a "channel creation" strategy $\mathsf{new} : I \rightarrow ch(A)$ (where I is the arena I with a single initial answer move). This responds to Opponent's initial question by generating a fresh name and making it public (i.e. playing an arbitrary move in N) — thus $\ker(\mathsf{new}) = \{\varepsilon, q\} \cup \{qi \mid i \in \mathbb{N}\}$. For each A-rooted arena A, we have a strategy $\mathsf{recv} : ch(A) \rightarrow A$ which responds to Opponent's initial question — which supplies the a channel name i — by playing copycat between A and the dependent arena of i. (See Fig. 1.)

To define the composition of strategies $\sigma : A \rightarrow B$ and $\tau : B \rightarrow C$ we need to allow interaction both in the shared "public arena" B and in the dependent arenas. In addition, we impose the "freshness conditions" introduced in [11] to ensure that the new names introduced by σ are disjoint from those introduced by τ, and that both are disjoint from those introduced by Opponent.

Definition 7. *An interaction sequence is a justified sequence t with two partial polarizations λ^L and λ^R such that:*

– *Every move has at least one polarity: for all $s \sqsubseteq t . \lambda^L(s) \downarrow$ or $\lambda^R(s) \downarrow$.*
– *There is no $s \sqsubseteq t$ such that $\lambda^L(s) = \lambda^R(s)$.*

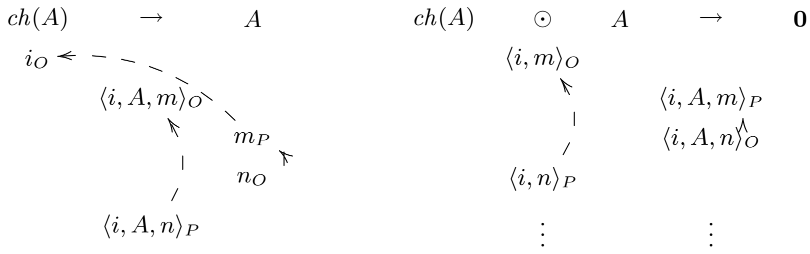


Fig. 1. Typical plays of recv and snd

– $P_\nu(s{\restriction}L) \cap P_\nu(s{\restriction}R) = (P_\nu(s{\restriction}L) \cup P_\nu(s{\restriction}R)) \cap O_\nu(s{\restriction}L\triangle R) = \varnothing$,

(Where we write $s{\restriction}L$ for the polarized sequence obtained by restricting to moves for which λ^L is defined, and $s{\restriction}L\triangle R$ for the restriction to moves for which only one of λ^L, λ^R is defined.)

Let $I_{A,B,C}$ be the set of interaction sequences which are legal sequences of $(A \to B) \to C$. Given $\sigma : A \to B, \tau : B \to C$, we may now define:
$\sigma;\tau : A \to C = \{s \in L^-_{A\to C} \mid \exists t \in I_{A,B,C}.t{\restriction}L \in \sigma \wedge t{\restriction}R \in \tau \wedge s = t{\restriction}L\triangle R\}$.

We prove that composition is well-defined and associative following the standard arguments used in [7, 15, 5], extended to ν-strategies in [11]. The identity strategy $\mathsf{id}_A : A \to A$ is defined: $\ker(\mathsf{id}_A) = \{s \in (L^-_{A\to A})^E \mid \forall t \sqsubseteq^E s.t{\restriction}A^+ = t{\restriction}A^-\}$. Thus we may form a category $\mathcal{G}$ in which objects are A-rooted tree arenas and morphisms from A to B are strategies on $A \to B$.

$\mathcal{G}$ has all small coproducts, given by the "disjoint union" of arenas, and an initial object, the empty arena $\mathbf{0}$, containing no moves. We define *premonoidal* structure on $\mathcal{G}$ based on that described in [5, 13, 11], in which initial moves of $A \odot B$ are pairs of initial moves from A and B, but non-initial moves are either from A or from B.

Definition 8. *From A-rooted tree arenas A_1, A_2, we form $A_1 \odot A_2$:*

– $M_{A_1\odot A_2} = \{(m,n) \in (M_{A_1} \times M^I_{A_2}) \cup (M^I_{A_1} \times M_{A_2}) \mid i \in \nu(m) \cap \nu(n) \Longrightarrow \alpha(m)_i = \alpha(n)_i)\}$,
– $\lambda_{A_1\odot A_2}(\langle m_1, m_2\rangle) = \lambda_{A_2}(m_2)$, *if* $m_1 \in M^I_{A_1}$,
  $\lambda_{A_1\odot A_2}(\langle m_1, m_2\rangle) = \lambda_{A_1}(m_1)$, *otherwise,*
– $(m_1, m_2) \vdash_{A_1\odot A_2} (n_1, n_2)$ *if* $m_1 = n_1 \in M^I_{A_1}$ *and* $m_2 \vdash_{A_2} n_2$ *or* $m_2 = n_2 \in M^I_{A_2}$ *and* $m_1 \vdash_{A_2} n_1$,
– $\pi \cdot \langle m, n\rangle = \langle \pi \cdot m, \pi \cdot n\rangle$,
– $\alpha(m,n)_i = \alpha^{A_1}(m)_i$, *if* $i \in \nu(m)$,
  $\alpha(m,n)_i = \alpha^{A_2}(n)_i$, *otherwise.*

Examples:

Equality Testing For each object A, we have a strategy $\mathsf{eq} : ch(A) \odot ch(A) \to I + I$ which is supplied by Opponent with a pair of names and responds with $\mathrm{in_l}(*)$ if they are equal and $\mathrm{in_r}(*)$ otherwise: $\ker(\mathsf{eq}) = \{\langle i,i\rangle \mathrm{in_l}(*) \mid i \in \mathbb{N}\} \cup \{\langle i,j\rangle \mathrm{in_r}(*) \mid i,j \in \mathbb{N} \wedge i \neq j\}$

Message Passing For each object A, we have a strategy $\mathsf{send} : ch(A) \odot A \to \mathbf{0}$ which is supplied with a channel name i and an initial move m in A, plays it as an initial move $\langle i, m\rangle$ in the dependent arena for A, and then plays copycat between the explicit and dependent occurrences of A. (See Fig. 1.)

For each object A we define an endofunctor $\_ \odot A : \mathcal{G} \to \mathcal{G}$: given $\sigma : B \to C$, $\sigma \odot A : B \odot A \to C \odot A = \{s \in L_{B\odot A \to C \odot A} \mid s\backslash A, A \in \sigma \wedge s{\restriction}A, A \in \mathsf{id}_A \wedge P_\nu(s\backslash A, A) = P_\nu(s)\}$.

Proposition 1. *$(\mathcal{G}, I, \odot)$ is a symmetric premonoidal category.*

We now identify a category in which the premonoidal product is Cartesian.

Definition 9. *A sequence $qas \in L_{A\to B}$ is* single-threaded *if a answers q and:*

- *Player does not introduce any new names with the move a — i.e $P_\nu(qa) = \varnothing$.*
- *There is at most one move justified by a in s.*

A strategy σ is single-threaded if it is non-empty, every sequence of at least two moves in $\ker(\sigma)$ is single-threaded and $qa, qa' \in \ker(\sigma)$ implies $a = a'$.

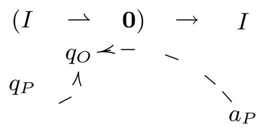
To define the composition of single-threaded strategies we apply a "promotion" operation $(\_)^\dagger$.

Definition 10. *Given a strategy $\sigma : A$, let $\sigma^\dagger$ be the least subset of L_A such that for any interaction sequence s, if $qa(s{\restriction}L) \in \sigma^\dagger$, $qa(s{\restriction}R) \in \sigma$ and $qa(s{\restriction}L\triangle R) \in L_A$ then $qa(s{\restriction}L\triangle R) \in \sigma^\dagger$.*

We define a category $\mathcal{G}_t$ with tree arenas as objects and single-threaded total strategies on $A \to B$ as morphisms from A to B. Composition of $\sigma : A \to B$ and $\tau : B \to C$ is defined $\sigma^\dagger;\tau$, and the identity on A is the restriction of $\mathsf{id}^{\mathcal{G}}_A$ to single-threaded sequences. We also note that $\_ \odot \_$ is a Cartesian product on $\mathcal{G}_t$. Thus $(\mathcal{G}, \mathcal{G}_t, (\_)^\dagger)$ is a Freyd category [16] (a symmetric premonoidal category $\mathcal{G}$, a Cartesian category $\mathcal{G}_t$, and an identity-on-objects strict symmetric premonoidal functor from $\mathcal{G}_t$ to $\mathcal{G}$). Moreover, it is a *closed* Freyd category: the functor $(\_)^\dagger \odot A : \mathcal{G}_t :\to \mathcal{G}$ has a right adjoint $A \rightharpoonup \_ : \mathcal{G} \to \mathcal{G}_t$.

Definition 11. *For any Q-rooted tree arena B, let $\uparrow B$ be the arena obtained by adding to B a single initial answer (invariant under G action) which enables all of the initial moves of B. We then define $A \rightharpoonup B = \uparrow (A \to B)$.*

Thus, for example, the arena $I \rightharpoonup \mathbf{0}$ consists of an initial answer which enables a single question. So we have a strategy $\mathsf{spawn} : (I \rightharpoonup \mathbf{0}) \to I$ which responds to the initial Opponent question by concurrently answering it and playing the (unique) initial question in $I \to \mathbf{0}$. i.e.:



There is an obvious bijection from (non-empty) legal sequences on $A \odot B \to C$ to single-threaded sequences on $A \to (B \rightharpoonup C)$, sending $\langle m,n\rangle s$ to $mans$ and yielding an adjunction between $A \rightharpoonup \_$ and $(\_)^\dagger \odot A : \mathcal{G}_t :\to \mathcal{G}$. Thus we have a model of the call-by-value λ-calculus, and so we may interpret terms $x_1 : S_1, \ldots, x_n : S_n \vdash M : T$ of $\mathcal{L}$ as morphisms from $[\![S_1]\!] \odot \ldots \odot [\![S_n]\!]$ to $[\![T]\!]$ in $\mathcal{G}$ by setting $[\![\mathsf{0}]\!] = \mathbf{0}$, $[\![\mathtt{unit}]\!] = I$ and $[\![\mathtt{bool}]\!] = I + I$, $[\![S \Rightarrow T]\!] = [\![S]\!] \rightharpoonup [\![T]\!]$, and $[\![\mathtt{chan}[T]]\!] = ch([\![T]\!])$. The constants are interpreted as the key strategies already defined for channel-generation, equality testing, thread-spawning and sending and receiving values.

In order to prove soundness and adequacy with respect to may-testing (i.e. $M \Downarrow$ if and only if $M \neq \bot$) we define the interpretation of configurations $(\mathcal{T}, \mathcal{N})$. Given strategies $\sigma : A \to \mathsf{0}$ and $\tau : A \to B$, we define the asymmetric interleaving $\sigma \| \tau : A \to B$ to consist of sequences $qs \in L_{A \to B}$ such that there exists an interaction sequence t with $q(t \upharpoonright L) \in \sigma$, $q(t \upharpoonright R) \in \tau$ and $q(t \upharpoonright L \triangle R) = qs$. Then $f \| (g; h) = (f \| g); h$.

We interpret the configuration $M_1 : \mathsf{0}, \ldots, M_n : \mathsf{0}, N : S, \{a_1 : \mathtt{chan}[T_1], \ldots, a_m : \mathtt{chan}[T_n]\}$ as $\mathsf{new}_{[\![T_1]\!]} \odot \ldots \mathsf{new}_{[\![T_m]\!]}; ([\![M_1]\!] \| \ldots \| ([\![M_n]\!] \| [\![N]\!]))$.

Lemma 1. *If $C \longrightarrow C'$ then $[\![C]\!] \subseteq [\![C']\!]$.*

Proof. We show that we may interpret evaluation contexts $a_1 : T_1, \ldots, a_n : T_n \vdash E[\cdot : S] : T$ as morphisms $[\![E[\_]]\!] : [\![S]\!] \odot [\![T_1]\!] \odot \ldots \odot [\![T_n]\!] \to [\![T]\!]$ so that $[\![E[M]]\!] = \delta_{[\![T_1]\!] \odot \ldots \odot [\![T_n]\!]}; ([\![M]\!] \odot ([\![T_1]\!] \odot \ldots \odot [\![T_n]\!])); [\![E[\_]]\!]$ and verify soundness for each reduction of the operational semantics using the categorical structure of $\mathcal{G}$ and the following (in)equations:

- For any $f : A \to \mathbf{0}$ and $g : I \to B$, $\Lambda(f); \mathsf{spawn}; g = f \| t_A; g$[1],
- $\pi_r \subseteq \mathsf{send} \| (\pi_l; \mathsf{recv})$,
- $\mathsf{new}_A; \langle \mathsf{id}_{ch(A)}, \mathsf{id}_{ch(A)}, \mathsf{id}_{ch(A)} \rangle^\dagger; \mathsf{eq} \odot ch(A) = \mathsf{new}_A; (\mathrm{in_l} \odot ch(A))$,
- $(\mathsf{new}_A \odot ch(A)); \langle \pi_l, \pi_r, \pi_l, \pi_r \rangle^\dagger; (\mathsf{eq} \odot (ch(A) \odot ch(A))) = (\mathsf{new}_A \odot ch(A)); \mathrm{in_r} \odot (ch(A) \odot ch(A))$.

Thus $M \Downarrow$ implies $M \neq \bot$.

Proposition 2. *If $[\![C]\!] \neq \bot$ then $M \Downarrow$.*

Proof. We define a translation which allows us to count internal reductions of M as $\mathtt{recv}$ actions: fixing a variable $c : \mathtt{chan}[\mathtt{unit}]$, for each term $\Gamma \vdash M : T$ with $c \notin \Gamma$, define $\Gamma, c \vdash M^c : T$:

- $M^c = M$ if M is a variable or constant.
- $(M\,N)^c = (\mathtt{recv}\,c); (M^c\,N^c)$
- $(\lambda x.M)^c = \lambda x.M^c$

Then for every term $\Gamma \vdash M : T$, $[\![M]\!] = [\![\nu c.!\mathtt{send}\,(c, ())|M^c]\!]$. Defining $\mathtt{send}^1\,M = \mathtt{send}\,M$ and $\mathtt{send}^{i+1}\,M = \mathtt{send}\,M|\mathtt{send}^i\,M$, we have $[\![!\mathtt{send}\,V]\!] = \bigcup_{i \in \mathbb{N}} [\![\mathtt{send}^i\,V]\!]$. So if $[\![M]\!] \neq \bot$ then by continuity, there exists n such that $[\![\mathtt{new}c.\mathtt{send}^n\,(c, ())|M^c]\!] \neq \bot$. We may prove by induction on n that this entails that $M \Downarrow$, by showing that for any configuration $[\![C]\!] = \bigcup\{[\![C']\!] \mid C \longrightarrow C'\}$.

[1] $t_A : A \to I$ is the terminal map in the category of single threaded strategies.

5 Definability and Full Abstraction

We prove full abstraction by establishing that for any legal sequence s over a type-object, the least strategy containing s (the closure of $\{s\}$ under the relations $\sqsubseteq$, $\sim$ and $\preceq$) is the denotation of a term. This is sufficient to define "tests" to distinguish any pair of distinct strategies.

Definition 12. *For any $s \in L^{-}_{A}$, let $\lceil s \rceil$ be the least set such that $s \in \lceil s \rceil$, and if $t \in \lceil s \rceil$ and $r \sqsubseteq t$ or $r \sim t$ or $r \preceq t$ then $r \in \lceil s \rceil$.*

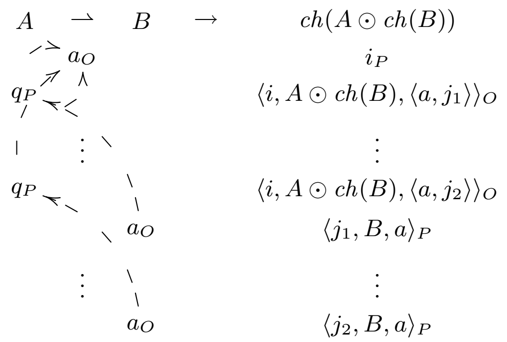


Fig. 2. A typical play of $[\![\mathtt{in}]\!](f)$

The key to our proof is the observation that we may encode justification pointers in terms of explicit name passing, and so reduce an arbitrary justfied sequence to one which is "pointer-free" (i.e. every move is an initial move). This reduction corresponds, in essence, to Sangiorgi's "triggering" translation of higher-order π-calculus into the π-calculus [18]. In the semantic setting, its essence may be captured as a *definable retraction* from the function type $S \Rightarrow T$ into the channel type $\mathtt{chan}[S * \mathtt{chan}[T]]$ (i.e. a pair of terms $(\mathtt{in} : (S \Rightarrow T) \Rightarrow \mathtt{chan}[S * \mathtt{chan}[T]], \mathtt{out} : \mathtt{chan}[S * \mathtt{chan}[T]] \Rightarrow S \Rightarrow T)$ such that $[\![\lambda x.\mathtt{in}\,(\mathtt{out}\,x)]\!] = [\![\lambda x.x]\!]$.

Lemma 2. *For any types S, T there is a definable retraction from $S \Rightarrow T$ to $\mathtt{chan}[S * \mathtt{chan}[T]]$.*

Proof. We have $\mathtt{in} = \lambda f.\nu c.(\mathtt{let}\ z = \mathtt{recv}\ c\ \ \mathtt{in}\ \ \mathtt{send}\,(\mathtt{snd}(z), f\,\mathtt{fst}(z)))|c$, $\mathtt{out} = \lambda x.\lambda y.\nu d.\mathtt{send}\,(x,(y,d))|\mathtt{recv}\,d$.
There is a translation from legal plays of $[\![S \Rightarrow T]\!]$ to ($\sim$-equivalence classes of) legal plays of $[\![\mathtt{chan}[S * \mathtt{chan}[T]]\!]$, which adds arbitrary names i, j to the opening pair of moves, and replaces each move m with a justification pointer into one of these moves with a dependent move of the form $\langle i, m \rangle$ or $\langle j, m \rangle$. We prove that $\mathtt{in}$ and $\mathtt{out}$ act as copycats factoring through this translation (see Fig. 3).

We cannot use this retraction to reduce definability at all types to behaviour at the "π-types" constructed from B, $\_*\_$ and $\mathtt{chan}[\_]$ (the problem is that $\mathtt{chan}[\_]$ is non-functorial and so "is a definable retract of" is not a precongruence). However, we may use it to map each strategy into one which has a "pointer-free fragment" from which the original strategy can be recovered. For each type T we define a type $\overline{T}$ as follows: $\overline{B} = B$; $\overline{S * T} = \overline{S} * \overline{T}$ and $\overline{S \Rightarrow T} = \mathtt{chan}[\overline{S} * \mathtt{chan}[\overline{T}]]$.

Proposition 3. *For each type T there is a definable retraction $(\mathtt{inj}_T, \mathtt{proj}_T) : T \trianglelefteq \overline{T}$ such that for all $s \in [\![T]\!] \rightarrow \mathbf{0}$ there exists a pointer-free $\overline{s} \in [\![x \vdash \mathtt{proj}_T\, x]\!]; \lceil s \rceil$ such that $[\![x \vdash \mathtt{inj}_T\, x]\!]; \lceil \overline{s} \rceil = \lceil s \rceil$.*

So given a sequence s in $[\![T]\!] \rightarrow \mathbf{0}$, if $\lceil \overline{s} \rceil$ is definable as a term $x : \overline{T} \vdash M : \mathtt{0}$ then $\lceil s \rceil$ is definable as $M\,(\mathtt{inj}_T\, x)$. We now show that each such pointer-free strategy is definable, via a decomposition which successively erases dependent moves.

Proposition 4. *For any pointer-free $s \in [\![T_1]\!] \odot \ldots \odot [\![T_n]\!] \rightarrow \mathbf{0}$, $\lceil s \rceil$ is definable as a term $x_1 : T_1, \ldots, x_n : T_n \vdash M : \mathtt{0}$ such that $[\![M]\!] = \lceil s \rceil$.*

Proof. We assume $T_1, \ldots, T_n$ are *pointed* (i.e. base, function or channel types) and define M by induction on the length of s. If this is less than 2, then $\lceil s \rceil = \bot = [\![\mathtt{nil}]\!]$.

If s has length greater than 2 then $s = \langle a_1, \ldots, a_n \rangle \langle i, B, b \rangle s'$, where $i \in \nu(\langle a_1, \ldots, a_n \rangle)$ and thus $i = a_j$ for some $1 \leq j \leq n$, and so $B = [\![T_j]\!]$. So if $T_j = \mathtt{chan}[S_1 * \ldots * S_m]$ (where each S_k is pointed) then $b = \langle b_1, \ldots, b_m \rangle$ where $b_k \in M^I_{[\![S_k]\!]}$ for each k. So we may form a legal sequence $s'' = \langle a_1, \ldots, a_n, b_1, \ldots, b_m \rangle s'$ on $[\![T_n]\!] \odot \ldots \odot [\![T_n]\!] \odot [\![S_1]\!] \odot \ldots \odot [\![S_m]\!]$. By induction hypothesis, $\lceil s'' \rceil$ is definable as a term $x_1 : T_1, \ldots, x_n : T_n, y_1 : S_1, \ldots, y_m : S_m \vdash M : \mathtt{0}$

If $\langle i, B, b \rangle$ is an Opponent move then we have $\lceil s \rceil = [\![\mathtt{let}\,(y_1, \ldots, y_m) = \mathtt{recv}\, x_i\ \mathtt{in}\ M]\!]$. If $\langle i, B, b \rangle$ is a Player move then for each $k \leq m$ we define a term $x_1 : T_1, \ldots, x_n : T_n \vdash N_k : S_k$:

- If $S_k = \mathtt{unit}$ then $N_k =_{df} ()$,
  If$S_k = \mathtt{bool}$ then $N_k =_{df} \mathtt{tt}$ if $b_k = \mathrm{in_l}(*)$ and $N_k =_{df} \mathtt{ff}$, otherwise.
- If $S_k = U \Rightarrow V$ then let $N_k =_{df} \lambda x.\mathtt{nil}$.
- If $S_k = \mathtt{chan}[U]$ then $N_k =_{df} x_i$, if $b_k = a_i$ and $b_k \neq x_j$ for $j < i$,
  $N_k =_{df} \mathtt{new}_u\,()$, if $b_k \neq x_j$ for all $j \leq n$.

For each $j \leq n$ we define a test term $B_j : \mathtt{bool}$:

- If $T_j = \mathtt{bool}$, we define $B_j =_{df} x_i$ if $a_j = \mathrm{in_l}(*)$ and $B_j =_{df} \neg x_i$, otherwise.
- If $T_j = U \Rightarrow V$ or $T_j = \mathtt{unit}$, then $B_j =_{df} \mathtt{tt}$,
- If $T_l = \mathtt{chan}[U]$ then $B_l = \bigwedge_{k \leq n} E_k$, where:
  $E_k =_{df} x_i = x_j$ if $T_j = T_k$ and $a_j = a_k$,
  $E_k =_{df} \neg x_j = x_k$ if $T_j = T_k$ and $a_j \neq a_k$,
  $E_k =_{df} \mathtt{tt}$, otherwise.

Then $\lceil s \rceil$ is definable as:
$\mathtt{let}\,(y_1, \ldots, y_m) = (N_1, \ldots, N_m)\ \mathtt{in\ If}\ \bigwedge_{j \leq n} B_j\ \mathtt{then}\,(\mathtt{send}\,(x_i, y_1, \ldots, y_m)|M)\ \mathtt{else\ nil}$.

Theorem 1. *$[\![M]\!] \subseteq [\![N]\!]$ if and only if $M \lesssim N$.*

Proof. From right-to left (inequational soundness) this follows from soundness and adequacy. We prove the converse for closed values U, V, which implies the general case. So suppose $[\![U]\!] \not\subseteq [\![V]\!]$. Then there exists a sequence $qs \in I \rightarrow [\![T]\!]$ such that $qs \in [\![U]\!]$ and $qs \notin [\![V]\!]$. By Propositions 3 and 4, the strategy $\lceil s^{\perp} * \rceil$ on $[\![T]\!] \rightarrow \texttt{unit}$ (where $*$ is the unique move in I) is definable as a term $x : T \vdash P : \texttt{unit}$. Then $[\![(\lambda x.P)\,U]\!] = \{*\}$ and hence by adequacy, $(\lambda x.P)\,U \Downarrow$. But $[\![\lambda x.M\,V]\!] = \perp$, since for all $t* \in \lceil s^{\perp} * \rceil$ there exists $r \sim s^{\perp}$ such that $t \preceq r$ and hence $s \sim r^{\perp} \preceq t^{\perp}$ and so by assumption $qt^{\perp} \notin [\![V]\!]$. Hence $(\lambda x.M)\,V \not\Downarrow$, and $U \not\lesssim V$ as required.

References

1. W. Ferreira, M. Hennessy, and A. S. A. Jeffrey. A theory of weak bisimulation for core CML. *J. Functional Programming*, 8(5):447–491, 1998.
2. D. Ghica and G. McCusker. The regular language semantics of second-order Idealised Algol. *Theoretical Computer Science*, 309:469 – 502, 2003.
3. D. Ghica and A. Murawski. Angelic semantics of fine-grained concurrency. In *Proceedings of FOSSACS '04*, number 2987 in LNCS, pages 211–225. Springer, 2004.
4. R. Harmer and G. McCusker. A fully abstract games semantics for finite nondeterminism. In *Proceedings of LICS '99*. IEEE Computer Society Press, 1998.
5. K. Honda and N. Yoshida. Game theoretic analysis of call-by-value computation. In *Proceedings of ICALP '97*, volume 1256 of *LNCS*. Springer-Verlag, 1997.
6. D. Hughes. Games and definability for System F. In *Proceedings of LICS '97*. IEEE Computer Society Press, 1997.
7. J. M. E. Hyland and C.-H. L. Ong. On full abstraction for PCF: I, II and III. *Information and Computation*, 163:285–408, 2000.
8. A. S. A. Jeffrey and J. Rathke. Contextual equivalence for higher-order pi-calculus revisited. In *Logical Methods in Computer Science* 1(1:4), 2002.
9. A. S. A. Jeffrey and J. Rathke. A fully abstract may-testing semantics for concurrent objects. In *Proceedings of LICS '02*, pages 101–112, 2002.
10. J. Laird. A game semantics of ICSP. In *Proceedings of MFPS XVII*, number 45 in Electronic notes in Theoretical Computer Science. Elsevier, 2001.
11. J. Laird. A game semantics of names and pointers. To appear in Annals of Pure and Applied Logic, available from http://www.cogs.susx.ac.uk/users/jiml, 2005.
12. J. Laird. A game semantics of the asynchronous pi-calculus. In *Proceedings of CONCUR '05*, number 3653 in LNCS, pages 51–65. Springer, 2005.
13. O. Laurent. Polarized games. In *Proceedings of LICS '02*, 2002.
14. P. Malacaria and C. Hankin. Generalised flowcharts and games. In *Proceedings of ICALP '98*, 1998.
15. G. McCusker. *Games and full abstraction for a functional metalanguage with recursive types*. PhD thesis, Imperial College London, 1996.
16. J. Power and E. Robinson. Premonoidal categories and notions of computation. *Mathematical Structures in Computer Science*, 1997.
17. J. Reppy. CML: A higher-order concurrent language. In *Proceedings of PLDI '91*, pages 293–305. ACM Press, 1991.
18. D. Sangiorgi. *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms*. PhD thesis, University of Edinburgh, 1993.