

A Game Semantics of Names and Pointers

J. Laird

Department of Informatics, University of Sussex, UK

Abstract

We describe a fully abstract semantics for a simple functional language with locally declared names which may be used as pointers to names. It is based on a category of dialogue games acted upon by the group of natural number automorphisms. This allows a formal, semantic characterization of key properties of names such as freshness and locality.

We describe a model of the call-by-value λ -calculus (a closed Freyd category) based on these games, and show that it can be used to interpret the *nu-calculus* of Pitts and Stark. We then construct a model of our pointer-language by extending our category of games with an explicit representation of the store, using a notion of semantic *garbage collection* to erase inaccessible pointers. Using factorization and decomposition techniques, we show that the compact elements of our model are definable as terms, and hence it is fully abstract.

Key words: Game Semantics, Fresh name generation, Pointer-based storage. AMS 68Q55

1 Introduction

Local names are a pervasive and subtle feature of higher-order programming languages and other calculi. Not only are they used for manipulating important constructs such as locally bound references and exceptions, name-passing is itself a very expressive computational paradigm, as demonstrated in particular by the π -calculus. Local names can also represent items of secret information which are dynamically generated, passed between agents and used to access further information or activity. They therefore have a key rôle in specifying properties of *secure* systems [1,30].

Email address: `jim1@sussex.ac.uk` (J. Laird).

¹ This research was supported by UK EPSRC grant GR/S72181.

Game semantics has proved successful in characterizing several programming features with locally declared names, including references [3–5], exceptions [15] and channels [16]. However, these models do not provide a direct representation of names: in the game semantics of references (for example), the `var` type is represented by an object with read and write “methods” which *might* behave like a true reference cell; new-name binding corresponds to composition with a strategy which actually does have such behaviour. Whilst generally successful, this indirect representation of names is inadequate to represent some important programming language features.

Good Variables All of the full abstraction results for the above mentioned models are contingent upon the presence of some form of “bad variables” (in the case of references, objects of reference type which do not behave correctly as storage cells). In general, extending with bad variables is not conservative with respect to observation equivalence, so models containing them are not fully abstract for reasoning about languages with only good variables, such as ML. ²

Pure Names Representing names via an object with an associated side effect (reference cell, channel etc.) fails when there is no such side effect. For example, the indirect approach fails to give a convincing semantic account of the *nu-calculus* of Pitts and Stark [25,28], which aims to capture the key features of names in a functional setting. One could argue that this failure is less significant if the object of modelling names is to understand the behaviour of e.g. named references, for which we do have a games model. However, pure names may also be used to represent secrets such as cryptographic keys for which we do not.

Pointers Existing methods can only model a reference to a reference as cell which can store another reference cell, rather than the natural representation of a pointer as a cell storing the *name* of another cell. Reasoning about the behaviour of pointers in higher-order settings is known to be difficult: in particular, the accessibility of pointers from different components of a program is hard to analyze, and a direct semantic representation could be the basis of model-checking techniques for doing so. Pointers of this kind also have much in common with other “self-referential” names such as channel names in the π -calculus.

In this paper, we develop a direct approach to modelling names in game semantics. We show how this can contribute in the areas described above, by describing a fully abstract semantics of a call-by-value language with pointers ($\lambda\nu!$) which has no bad variables. More generally, it shows how interaction can be combined with a side-effect (state-transformation) to describe programs which use the store in a more “extensional” way.

² McCusker has shown that Idealized Algol with active expressions is an exception [21].

We also give a simple games model of the nu-calculus: although this is not fully abstract, it does give a denotational characterization of the equivalence arising when names are allowed to leak to environment through the store (as in ML, for instance). This illustrates the observation (see e.g. [13]) that allowing leakage of names in this way simplifies reasoning about contextual equivalence.

1.1 Organization of the Paper

In Section 2 we describe the syntax and operational semantics of $\lambda\nu!$, a call-by-value functional language with locally declared pointers (essentially an extension of the nu-calculus with assignment of names to names, and dereferencing).

In Section 3, we describe the basis of our model, a category of (Hyland-Ong style) games and strategies, acted on by the automorphism group of natural numbers, from which we generate an equivalence corresponding to invariance under name substitution. In the setting of game semantics, of which a key feature is the distinction between Player (representing a program) and Opponent (representing the environment), new issues become evident: which participant in a dialogue introduced a given name, and how can knowledge of it pass from one to the other? We might also see this as reflecting the distinctions between *public* versus *private* knowledge. We use our category of games (and constructions for interpreting call-by-value function types developed by Honda and Yoshida [11]) to give a semantics for the nu-calculus.

In Section 4 we extend our notion of “games with names” so that it can be used to interpret $\lambda\nu!$. This is achieved by adding an explicit “store” to each move — a collection of pairs of names (n, m) , each representing a location and its contents (or the source and target of a pointer). This allows us to extend our model of the nu-calculus with simple and natural interpretations of assignment and dereferencing. In order to give a fully abstract model, we also need to eliminate inaccessible (and hence unobservable) pointers, and to do so we define a notion of “garbage collection” which exploits the fact that we can make a distinction between *globally accessible* names (which have been revealed to the environment) and *local* names (which have not).

In Section 5 we prove (using the game semantics techniques of *decomposition* and *factorization*) that all of the compact elements of our semantics are definable as terms of $\lambda\nu!$ extended with a single constant for divergence. We use this result to show that (when strategies are reduced to their sets of complete plays) our model of $\lambda\nu!$ is *fully abstract*.

1.2 Related Work

Much of the material in this paper appeared in preliminary form as an extended abstract in [17], where a game semantics of $\lambda\nu!$ was constructed, and shown to have a finite definability property. The major difference is that in the earlier work, the store is modelled as a separate part of the arena, with moves corresponding to assignment and dereferencing. In some ways, this is a simpler and more general approach. However, the resulting model is only fully abstract after a quotienting operation, for which an effective characterization is possible, but rather complicated. Here, we describe a more implicit interpretation of the store, which yields a direct presentation of the fully abstract model of $\lambda\nu!$.

The representation of names using the automorphism group of the natural numbers is already present in Stark’s model of the nu-calculus in the category of continuous G-sets [28]. Our definitions can thus be couched in the terminology of FM (Fraenkel-Mostowski) set theory, following the work of Pitts and Gabbay [8]. The extension of our model to capture pointers is reminiscent of Stark’s construction of a model of Reduced ML from a model of the nu-calculus [28], and bears out his observation that “dynamically created names really do capture the difficult part of ... references; actual value storage is not so hard”.

Abramsky, Ghica, Murawski, Ong and Stark have subsequently described a game semantics of the nu-calculus [2] based on a category of “nominal games” which is in many respects similar to the model described here. The main difference is that in [2] names are declared explicitly, allowing their scope to be determined and leading to a definability result for the nu-calculus. The technical apparatus of ν -arenas and strategies has recently been used by author to give a semantics of *higher-order concurrency* [18]. Names are used to represent channels on which interaction may occur, and which themselves may be passed as data values.

2 $\lambda\nu!$ — a Calculus of Names and Pointers

We study locally declared pointers in the setting of a call-by-value functional language, the $\lambda\nu!$ calculus. This is an extension of the nu-calculus with facilities for writing to and reading from names. Formally, $\lambda\nu!$ is a simply-typed λ -calculus, with types generated from the two ground types o (booleans) and ν (names), and the following constants:

Truth Values $\text{tt}, \text{ff} : o$

Conditional $\text{If} : o \Rightarrow T \Rightarrow T \Rightarrow T$
Equality Testing (for names) $\text{eq} : \nu \Rightarrow \nu \Rightarrow o$
New Name Declaration $\text{new} : \nu$
Assignment $\text{assign} : \nu \Rightarrow \nu \Rightarrow o$
Dereferencing $\text{deref} : \nu \Rightarrow \nu$.

We could, of course, include more types (e.g. the natural numbers), and features such as recursion, and it is straightforward to extend our semantics (along the lines of [7,12] etc.) to accommodate these features. It would also be fairly straightforward to give a model of **null** pointers, but instead we simply make it a convention that pointers point to themselves when declared.

We use the syntactic sugar $\nu x.M$ for $(\lambda x.M) \text{new}$, $M = N$ for $(\text{eq } M) N$, $M := N$ for $(\text{assign } M) N$, $!M$ for $\text{deref } M$ and $\text{If } M \text{ then } N_1 \text{ else } N_2$ for $((\text{If } M) \lambda x.M) \lambda x.N$ **tt**. Whilst $\lambda \nu!$ does not contain boolean-valued references, these can be simulated — for example, we may use a reference which points to itself as a reference to **tt** and a reference which points to some other arbitrary (freshly created) value as a reference to **ff**.

2.1 Operational Semantics

Let $\lambda \nu!^*$ be the extension of $\lambda \nu!$ with a set of constants $\{i : \nu \mid i \in \mathbb{N}\}$ (location names). Given a program M (a closed term of $\lambda \nu!^*$) we write $k \vdash M$ if k is a natural number such that every name in M is strictly less than k . A program-in-environment M, \mathcal{E} is a program M together with a pair $\mathcal{E} = (k, \mathcal{S})$ such that $k \vdash M$, and \mathcal{S} is an endofunction on $\{i \in \mathbb{N} \mid i < k\}$. We write $\mathcal{S}[i \mapsto j]$ for the function which maps i to j and is otherwise the same as \mathcal{S} .

The values of $\lambda \nu!^*$ are given by the following grammar:

$$V ::= i \mid \lambda x.N \mid C \mid \text{eq } V \mid \text{assign } V$$

where i ranges over all names, and C over all constants except **new**. The “big step” evaluation rules for evaluating a program and an environment to a value and an environment are shown in Table 1. We write $M \Downarrow V$ if $M, (0, \emptyset) \Downarrow V, \mathcal{E}$ for some \mathcal{E} . This is conservative over the operational semantics of the nu-calculus. It is also terminating — i.e. for every M there exists V such that $M \Downarrow V$, a fact which may be proved using a Tait-style reducibility predicate method. We adopt the same definition of contextual equivalence as for the nu-calculus: given terms $M, N : T$, $M \approx N$ if for all closing contexts $C[-] : o$, $C[M] \Downarrow \text{tt}$ if and only if $C[N] \Downarrow \text{tt}$.

Contextual equivalence in $\lambda \nu!$ is not conservative over the nu-calculus, as we may show using an example of an equivalence from [28] which is broken in $\lambda \nu!$. In the nu-calculus, $\lambda f : \nu \Rightarrow o. \text{tt} \approx \nu x. \nu y. \lambda f : \nu \Rightarrow o. (f x = f y)$, since to any argument to which we apply the latter term, the names replacing

$\overline{V, \mathcal{E} \Downarrow V, \mathcal{E}}$	$\overline{\text{new}, (n, \mathcal{S}) \Downarrow n, (n+1, \mathcal{S}[n \mapsto n])}$
$\frac{M, \mathcal{E} \Downarrow \lambda x. M', \mathcal{E}' \quad N, \mathcal{E}' \Downarrow V, \mathcal{E}'' \quad M'[V/x], \mathcal{E}'' \Downarrow U, \mathcal{E}'''}{M N, \mathcal{E} \Downarrow U, \mathcal{E}'''}$	
$\frac{M, \mathcal{E} \Downarrow i, \mathcal{E}' \quad N, \mathcal{E}' \Downarrow i, \mathcal{E}''}{M = N, \mathcal{E} \Downarrow \text{tt}, \mathcal{E}''}$	$\frac{M, \mathcal{E} \Downarrow i, \mathcal{E} \quad N, \mathcal{E}' \Downarrow j, \mathcal{E}''}{M = N, \mathcal{E} \Downarrow \text{ff}, \mathcal{E}''} \quad i \neq j$
$\frac{M, \mathcal{E} \Downarrow \text{tt}, \mathcal{E}'}{\text{If } M, \mathcal{E} \Downarrow \lambda xy. x, \mathcal{E}'}$	$\frac{M, \mathcal{E} \Downarrow \text{ff}, \mathcal{E}'}{\text{If } M, \mathcal{E} \Downarrow \lambda xy. y, \mathcal{E}'}$
$\frac{M, \mathcal{E} \Downarrow i, \mathcal{E}' \quad N, \mathcal{E}' \Downarrow j, (k, \mathcal{S})}{M := N, \mathcal{E} \Downarrow \text{tt}, (k, \mathcal{S}[i \mapsto j])}$	$\frac{M, \mathcal{E} \Downarrow i, (k, \mathcal{S})}{!M, \mathcal{E} \Downarrow j, (k, \mathcal{S})} \quad \mathcal{S}(i) = j$

Table 1

Operational Semantics of $\lambda\nu!$

x and y will be fresh and hence any test on them will produce the same result. However, they can be distinguished in $\lambda\nu!$, using a context which simply observes whether the argument is applied to anything: for example, $\nu z. z := z; ([\cdot](\lambda n. z := n)); !z = z$ (which returns tt in the first case and ff in the second).

3 Games with Names

Our notion of game is based on the dialogue games of Hyland and Ong [12] (and Nickau [24]), to which we add structure for manipulating a countable set of names, in the form of an action of the automorphism group of the natural numbers. This generates an equivalence on strategies corresponding to invariance under substitution of names. (A similar equivalence is used in [7], to preserve parametricity in games with a countable set of indices for threads: the key difference is that these indices cannot be passed between strategies as names can, and so the induced equivalence can be described componentwise, as it is in [7].) We give brief (and slightly non-standard) definitions of arenas and legal sequences and refer the reader to the literature ([12,20,14,11] etcetera) for more detailed explanation.

An (underlying) arena A is a tuple $(M_A, M_A^I, \lambda_A, \vdash_A)$ consisting of a set of moves M_A , a subset $M_A^I \subseteq M_A$ of *initial* moves, a question/answer labelling $\lambda_A : M_A \rightarrow \{Q, A\}$ and an *enabling relation* $\vdash_A \subseteq M_A \times (M_A - M_A^I)$ such that no answer is enabled by an answer, and we may partition moves into two polarities (“Player” and “Opponent”) by requiring that every initial move is an Opponent move, and every move enabled by an Opponent move is a Player

move and vice-versa. We describe an arena as A -rooted if all of its initial 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. We may represent these pointers as a partial function $j : \mathbb{N} \rightarrow \mathbb{N}$ such that $j(i) = k$ if the i th move in s is justified by the k th move. A justified sequence is *alternating* if Opponent moves are followed by Player moves and vice-versa, and *well-opened* if there is at most one initial move (the first move). We write J_A for the set of finite, alternating and well-opened justified sequences of A .

Definition 3.1 We define the view³ [12,20]: $\ulcorner s \urcorner$ of $s \in J_A$:
 $\ulcorner \varepsilon \urcorner = \varepsilon$, $\ulcorner m \urcorner = m$ and $\ulcorner smtn \urcorner = \ulcorner s \urcorner mn$ if m justifies n .

The set L_A of legal sequences of A consists of the sequences $s \in J_A$ which satisfy the following conditions:

Well-bracketing The justifier of each answer is the most recent unanswered question.

Visibility If $ta \sqsubseteq s$ then the justifier of a appears in $\ulcorner t \urcorner$.

3.1 ν -Arenas

Let G be Baire Space — the topological group of automorphisms on \mathbb{N} with the product topology on $\mathbb{N}^{\mathbb{N}}$, for which a basis of neighbourhoods of the identity is $\{\text{stab}_G(k) \mid k \subseteq^{fin} \mathbb{N}\}$ [28]. By a continuous action of G upon a set A , we will mean a G -action which is continuous with respect to the discrete topology on A . So the stabiliser of any element $a \in A$ is open in G and there is a finite subset $k \subseteq \mathbb{N}$ such that if $\pi \in \text{stab}_G(k)$ then π is in the stabilizer of a . We write $\nu(a)$ (the *support* of a) for the smallest such subset. In other words, A is a FM (Fraenkel-Mostowski) set [8] — a set with an action of G upon it such that every element of A has finite support.

Definition 3.2 A ν -arena is an arena A and a continuous action of G on M_A such that $\pi \cdot m \in M_A^I$ iff $m \in M_A^I$, $\lambda_A(\pi(m)) = \lambda_A(m)$ and $m \vdash n$ iff $\pi(m) \vdash \pi(n)$. (In other words, M_A^I is a FM-predicate, λ_A a FM-function and \vdash_A a FM-relation.)

Every arena gives rise to a ν -arena in which the group action is the identity on all moves. A slightly less trivial example is the ν -arena N , which will be used to interpret the type ν . This has as its set of moves the natural numbers

³ If Player (resp. Opponent) is about to move, then $\ulcorner s \urcorner$ is the usual Player (resp. Opponent) view.

(all of which are initial answers), 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 $\pi \cdot i = \pi(i)$.

By preservation of the labelling and enabling relation on moves we obtain the following fact.

Lemma 3.3 *For any ν -arena A there is a continuous action of G on L_A defined $\pi \cdot (m_1 m_2 \dots m_n) = (\pi \cdot m_1)(\pi \cdot m_2) \dots (\pi \cdot m_n)$.*

We write $\nu(s)$ for the support of the sequence s . We define functions $P_\nu, O_\nu : L_A \rightarrow \mathcal{P}_{fin}(\mathbb{N})$ which partition $\nu(s)$ into the sets of new names introduced by Player and (respectively) Opponent in s .

Definition 3.4 *The functions P_ν, O_ν are defined:*

$$\begin{aligned} P_\nu(\varepsilon) &= \emptyset, \\ P_\nu(sa) &= P_\nu(s) \cup (\nu(sa) - \nu(s)) \text{ if } a \text{ is Player move,} \\ P_\nu(sa) &= P_\nu(s) \text{ otherwise.} \end{aligned}$$

$$O_\nu(s) = \nu(s) - P_\nu(s).$$

We write \sim for the equivalence relation on justified sequences determined by the orbits of the group action — i.e. $s \sim t$ if $\exists \pi \in G. \pi \cdot s = t$.

A *strategy* on a ν -arena is, in essence, the orbit of a deterministic strategy on the underlying arena (or, more precisely, the union of the orbits of the legal sequences of a strategy on the underlying arena).

Definition 3.5 *Let (A, \cdot) be a ν -arena. A ν -strategy $\sigma : A$ is a non-empty and even-prefix-closed set of even-length legal sequences of A satisfying the following conditions:*

\sim -saturation If $s \in \sigma$ and $s \sim t$ then $t \in \sigma$.

\sim -determinacy If $sa, tb \in \sigma$ and $s \sim t$ then $sa \sim tb$.

Equivalently, if $sa \in \sigma$ and $s \sim t$ then $tb \in \sigma$ if and only if $sa \sim tb$.

So, in particular, ν -strategies are not deterministic — when introducing a new name, Player must choose one of an infinite set of names which have not yet been used. Instead, we have \sim -determinacy, which says that Player's choice of move is deterministic with respect to the names which have been used, or deterministic up to renaming.

3.2 A Call-by-Value Category of Games

We will now construct a *premonoidal* category [27] of games in which to model the call-by-value λ -calculus aspect of $\lambda\nu!$. We follow essentially the constructions of Honda and Yoshida [11] or variants described by Laurent [19]. In each case the group action on compound arenas is defined pointwise. The key novelty thus lies in the definition of *composition* of strategies, since we must maintain distinctness of fresh names. (To adapt the call-by-value semantics of Abramsky and McCusker [4], which is based on a category $\text{Fam}(\mathcal{G})$ of indexed families of arenas, is somewhat complicated, since it requires the group action to be defined on both the indexing sets and their elements.)

Definition 3.6 *Given ν -arenas A_1, A_2 we define the (Q -rooted) call-by-value function-space ν -arena $A_1 \rightarrow A_2$ as follows:*

- $M_{A_1 \rightarrow A_2} = M_{A_1} + M_{A_2}$,
- $\lambda_{A_1 \rightarrow A_2}(\text{in}_i(m)) = Q$, if $i = 1$ and $m \in M_{A_1}^I$,
 $\lambda_{A_1 \rightarrow A_2}(\text{in}_i(m)) = \lambda_{A_i}(m)$, otherwise,
- $M_{A_1 \rightarrow A_2}^I = \text{in}_1(M_{A_1})$,
- $\vdash_{A_1 \rightarrow A_2} = \{ \langle \text{in}_i(m), \text{in}_i(n) \rangle \mid i \in \{1, 2\} \wedge \langle m, n \rangle \in \vdash_i \} \cup (\text{in}_1(M_{A_1}^I) \times \text{in}_2(M_{A_2}^I))$
- $\pi \cdot \text{in}_i(m) = \text{in}_i(\pi \cdot m)$.

This is similar to the call-by-name function-space arena of [12] except that play starts on the left — the initial moves in $A_1 \rightarrow A_2$ are the initial moves from A_1 relabelled as questions (to which the initial moves from A_2 are the answers). As an example, we consider the strategy $\mathbf{new} : I \rightarrow N$ (where I is the arena I with a single initial answer move) with which we shall interpret the declaration $\mathbf{new} : \nu$. This may respond to Opponent’s initial question with any move in N (representing a fresh name) — i.e. $\mathbf{new} = \{\varepsilon\} \cup \{qi \mid i \in \mathbb{N}\}$. This is the only non-empty strategy on $I \rightarrow N$.

Composition of strategies $\sigma : A \rightarrow B$ and $\tau : B \rightarrow C$ is by allowing interaction in B which is then hidden. This follows [6,12] although because of the way the call-by-value function-space is defined, composition starts on the left, with σ . More significantly, further conditions are required to ensure that the new names introduced by σ are disjoint from those introduced by τ , *and* from those introduced by Opponent in C (and that those introduced by τ are disjoint from those introduced by Opponent in A).

Definition 3.7 *The set of well-formed interaction sequences $I_{A,B,C}$ is the set of justified sequences s over the arena $(A \rightarrow B) \rightarrow C$ which satisfy $s \upharpoonright A, B \in L_{A \rightarrow B}$, $s \upharpoonright A, C \in L_{A \rightarrow C}$ and $s \upharpoonright B, C \in L_{B \rightarrow C}$, together with the following “freshness conditions”:*

$$i \ P_\nu(s \upharpoonright A \rightarrow B) \cap P_\nu(s \upharpoonright B \rightarrow C) = \emptyset,$$

ii $(P_\nu(s \upharpoonright A \rightarrow B) \cup P_\nu(s \upharpoonright B \rightarrow C)) \cap O_\nu(s \upharpoonright A \rightarrow C) = \emptyset$.

Given $\sigma : A \rightarrow B, \tau : B \rightarrow C$, we may now define:

$\sigma \upharpoonright \tau = \{s \in I_{A,B,C} \mid s \upharpoonright A, B \in \sigma \wedge s \upharpoonright B, C \in \tau\}$

and $\sigma; \tau : A \rightarrow C = \{s \in L_{A \rightarrow C} \mid \exists t \in \sigma \upharpoonright \tau. s = t \upharpoonright A, C\}$.

To prove that composition is well-defined, we use the freshness assumptions on interaction sequences to show that when σ introduces a new name in $s \upharpoonright (A \rightarrow B)$ then it is genuinely new in s (and similarly for τ) and also that when Opponent introduces a new name in $s \upharpoonright (A \rightarrow C)$ then it is new in s .

Lemma 3.8 *If $f : \mathbb{N} \rightarrow \mathbb{N}$ is a partial injection such that $\{x \in \mathbb{N} \mid f(x) \downarrow\}$ is finite, then there exists $\underline{f} \in G$ such that $f \subseteq \underline{f}$.*

PROOF: We define \underline{f} by induction as follows:

$\underline{f}(n) = f(n)$, if $f(n) \downarrow$,

$\underline{f}(n) = \min\{m \in \mathbb{N} \mid \forall i. (i < n \Rightarrow m > \underline{f}(i)) \wedge f(i) \downarrow \Rightarrow f(i) \neq m\}$, otherwise. \square

Lemma 3.9 *Suppose we have sequences sa, tb and $\pi, \pi' \in G$ such that $\pi \cdot s = t$, $\pi' \cdot a = b$, $\pi(n) = \pi'(n)$ for $n \in \nu(s) \cap \nu(a)$, and $\pi(n) \neq \pi'(m)$ for $n \in \nu(s)$ and $m \in \nu(sa) - \nu(s)$. Then $sa \sim tb$.*

PROOF: We define an automorphism $\pi'' : \mathbb{N} \rightarrow \mathbb{N} = \underline{f}$, where $f : \mathbb{N} \rightarrow \mathbb{N}$ is a partial injection defined as follows:

$f(n) = \pi(n)$ if $n \in \nu(s)$,

$f(n) = \pi'(n)$, if $n \in \nu(a)$.

By Lemma 3.8, π'' is a well-defined automorphism since f is a partial injection (defined only on $\nu(sa)$, which is finite): if $m \neq n$ then if $n, m \in \nu(s)$ then $f(m) = \pi(m) \neq f(n) = \pi(n)$ and if $n, m \in \nu(sa) - \nu(s)$ then $f(m) = \pi'(m) \neq f(n) = \pi'(n)$, and if $m \in \nu(s)$ and $n \in \nu(sa) - \nu(s)$ then $f(m) = \pi(m) \neq f(n) = \pi'(n)$ by assumption.

We have $\pi''(n) = \pi(n)$ for all $n \in \nu(s)$ and $\pi''(n) = \pi'(n)$ for $n \in \nu(a)$, and thus $\pi'' \cdot sa = (\pi'' \cdot s)(\pi'' \cdot a) = (\pi \cdot s)(\pi' \cdot a) = tb$ as required. \square

Lemma 3.10 *Suppose $sa, tb \in I_{A,B,C}$ and $s \sim t$. If:*

- *a is a Player move in $A \rightarrow B$ such that $sa \upharpoonright A \rightarrow B \sim tb \upharpoonright A \rightarrow B$ or*
- *a is a Player move in $B \rightarrow C$ such that $sa \upharpoonright B \rightarrow C \sim tb \upharpoonright B \rightarrow C$ or*
- *a is an Opponent move in $A \rightarrow C$ such that $sa \upharpoonright A \rightarrow C \sim tb \upharpoonright A \rightarrow C$*

then $sa \sim tb$.

PROOF: We prove this for the case in which a (and thus also b) is a Player move in $A \rightarrow B$. Assuming automorphisms $\pi, \pi' \in G$ such that $\pi \cdot s = t$ and

$\pi' \cdot (sa \upharpoonright A \rightarrow B) = tb \upharpoonright A \rightarrow B$ (and so in particular, $\pi' \cdot a = b$) we show that π, π' satisfy the requirements of Lemma 3.9, and thus $sa \sim tb$.

First, we show that if $n \in \nu(a)$ occurs in $\nu(s)$ then $n \in \nu(s \upharpoonright A \rightarrow B)$ (and so if $n \in \nu(a) \cap \nu(s)$ then $n \in \nu(s \upharpoonright A \rightarrow B)$ and $\pi(n) = \pi'(n)$ as required). Suppose for a contradiction that $n \notin \nu(s \upharpoonright A \rightarrow B)$ and so $n \in P_\nu(sa \upharpoonright A \rightarrow B)$. Then $n \notin P_\nu(sa \upharpoonright B \rightarrow C)$ by freshness property (i) and $n \notin O_\nu(sa \upharpoonright A \rightarrow C)$ by freshness property (ii) and hence in particular $n \notin \nu(sa \upharpoonright C)$. But then $n \notin \nu(s) = \nu(s \upharpoonright A \rightarrow B) \cup \nu(sa \upharpoonright C)$ which is the required contradiction.

Now suppose $m \in \nu(s)$ and $n \in \nu(a)$, and $\pi(m) = \pi'(n)$. Then $\pi'(n) \in \nu(t) \cap \nu(b)$, and so applying the above argument, $\pi'(n) \in \nu(t \upharpoonright A \rightarrow B)$ — i.e. $\pi'(n) = \pi'(l)$ for some $l \in \nu(s \upharpoonright A \rightarrow B)$, and so by injectivity of π' , $n \in \nu(s \upharpoonright A \rightarrow B) \subseteq \nu(s)$. So $m \in \nu(s)$ and $n \in \nu(sa) - \nu(a)$ implies $\pi(m) \neq \pi'(n)$ as required.

The case in which a is a Player move in $B \rightarrow C$ is precisely similar. If a is an Opponent move in $A \rightarrow C$ then the argument is similar, except that to show that if $n \in \nu(a)$ is not in $s \upharpoonright A \rightarrow C$ then it is not in s , we simply observe that if $n \in O_\nu(sa \upharpoonright A \rightarrow C)$ then $n \notin P_\nu(sa \upharpoonright A \rightarrow B) \cup P_\nu(sa \upharpoonright B \rightarrow C)$ and hence $n \notin \nu(sa \upharpoonright B)$. So $n \notin \nu(s)$ as required. \square

We write $s \preceq t$ if there exists $\pi \in G$ such that $\pi \cdot s \sqsubseteq t$.

Lemma 3.11 *If $s, t \in \sigma \upharpoonright \tau$, $s' \sqsubseteq s$ and $s' \upharpoonright A \rightarrow C \preceq t \upharpoonright A \rightarrow C$ then $s' \preceq t$.*

PROOF: By induction on the length of s' . Suppose $s'a \sqsubseteq s$, then by induction hypothesis there exists $t' \sqsubseteq t$ such that $s' \sim t'$. If a is a Player move in $A \rightarrow B$ then by \sim -determinacy of σ there exists b such that $t'b \sqsubseteq t$ and $s'a \upharpoonright A \rightarrow B \sim t'b \upharpoonright A \rightarrow B$. Similarly, if a is a Player move in $B \rightarrow C$ then $s'a \upharpoonright B \rightarrow C \sim t'b \upharpoonright B \rightarrow C$. In either case $s'a \sim t'b$ by Lemma 3.10.

If a is not a Player move in $A \rightarrow B$ or $B \rightarrow C$ then it is an Opponent move in $A \rightarrow C$. Then there exists b such that $t'b \sqsubseteq t$ and $s'a \upharpoonright A \rightarrow C \sim t'b \upharpoonright A \rightarrow C$, and so by Lemma 3.10, $s'a \sim t'b$ as required. \square

Proposition 3.12 *The composition of ν -strategies is a well-defined ν -strategy.*

PROOF: For \sim -saturation, suppose $s \in \sigma; \tau$ and $s \sim t$. Then there exists $s' \in \sigma \upharpoonright \tau$ such that $s = s' \upharpoonright A \rightarrow C$, and $\pi : \mathbb{N} \rightarrow \mathbb{N}$ such that $\pi \cdot s = t$. Then $\pi \cdot s' \in \sigma \upharpoonright \tau$ and hence $t = (\pi \cdot s') \upharpoonright A, C \in \sigma; \tau$ as required.

For \sim -determinacy, suppose $sa, tb \in \sigma; \tau$ and $s \sim t$. Then there exist sequences $s'a, t'b \in \sigma \upharpoonright \tau$ such that $s = s' \upharpoonright A \rightarrow C$ and $t = t' \upharpoonright A, C$. By Lemma 3.11 $s' \preceq t'b$ and $t' \preceq s'a$ and hence $s' \sim t'$. Suppose (w.l.o.g.) that the last move in s' (and hence also the last move in t') is an Opponent move in $A \rightarrow B$. Then a and b are

both moves in A and hence by \sim -determinacy of σ , $s'a \downarrow A \rightarrow B \sim t'b \downarrow A \rightarrow B$. So by Lemma 3.10, $s'a \sim t'b$ and hence $sa \sim tb$ as required. \square

As a simple example, we may consider how composition with the strategy **new** may be used to declare a new name. First, we define the “lifted function-space arena” with which we shall interpret the call-by-value function type (we shall consider its categorical properties later).

Definition 3.13 *Given a Q -rooted arena B , let $\uparrow B$ be the A -rooted arena obtained by adding to B a single initial answer (invariant under G action) which enables all of the initial moves of B — i.e.*

- $M_{\uparrow A} = \{a\} + M_A$,
- $M_{\uparrow A}^I = \{\text{in}_1(a)\}$,
- $\lambda_{\uparrow A} = [\{(a, A)\}, \lambda_A]$,
- $\vdash_{\uparrow A} = \{(\text{in}_1(m), \text{in}_r(n)) \mid (m = a \wedge n \in M_A^I) \vee \langle m, n \rangle \in \vdash_A\}$,
- $\pi \cdot \text{in}_1(a) = \text{in}_1(a), \pi \cdot \text{in}_r(m) = \text{in}_r(\pi \cdot m)$.

We then define the A -rooted arena $A \rightarrow B = \uparrow (A \rightarrow B)$

Let B be the arena with two answers tt, ff (with which we interpret the type o). Suppose we have an “equality test” strategy $\text{eq} : N \rightarrow (N \rightarrow B)$ consisting of even-prefixes of sequences of the form $iajtt$ (where $i = j$) and $iajff$ for $i \neq j$) — i.e. Opponent declares a name in N , Player replies with the initial answer $\text{in}N \rightarrow B$, Opponent plays another name (in $N \rightarrow B$ and Player returns tt if they are equal and ff otherwise. In any interaction sequence $s \in I_{I, N, N \rightarrow B}$ between $\text{new} : I \rightarrow N$ and eq the names i and j must be distinct, since i is introduced by Player in $I \rightarrow N$ and j by Opponent in $N \rightarrow (N \rightarrow B)$. In other words, the maximal sequences in $\text{new}|\text{eq}$ have the form $qiajff$, where $i \neq j$ and so $\text{new}; \text{eq} : I \rightarrow A = \{s \sqsubseteq qiajff \mid i \in \mathbb{N}\}$. Note that without the freshness conditions, this composite would fail to satisfy \sim -determinacy.

To show that we may form a category of ν -arenas we now need to show that composition is an associative operation on ν -strategies. First, we define two distinct ternary “parallel composition plus hiding” operations, corresponding to the two possible bracketings for binary composition.

Definition 3.14 *Given $\rho : A \rightarrow B, \sigma : B \rightarrow C, \tau : C \rightarrow D$, let $(\rho|\sigma)|\tau = \{s \in J_{((A \rightarrow B) \rightarrow C) \rightarrow D} \mid s \downarrow A, B, C \in \rho|\sigma \wedge s \downarrow A, C, D \in (\rho; \sigma)|\tau\}$ and $\rho|(\sigma|\tau) = \{s \in J_{(A \rightarrow (B \rightarrow C)) \rightarrow D} \mid s \downarrow B, C, D \in \sigma|\tau \wedge s \downarrow A, B, D \in \rho|(\sigma; \tau)\}$*

Lemma 3.15 *$s \in (\rho; \sigma); \tau$ if and only if there exists $t \in (\rho|\sigma)|\tau$ such that $s = t \downarrow A, D$. $s \in \rho; (\sigma; \tau)$ if and only if there exists $t \in \rho|(\sigma|\tau)$ such that $s = t \downarrow A, D$.*

PROOF: Suppose $s \in (\rho; \sigma); \tau$: then there exists $s' \in (\rho; \sigma)|\tau$ such that $s = s' \downarrow$

A, D and $s'' \in \rho|\sigma$ such that $s''\upharpoonright A, C = s'\upharpoonright A, C$. Then s', s'' may be (uniquely) written in the forms $x_1y_1u_1x_2y_2u_2 \dots x_my_mu_m$ and $x_1v_1y_1x_2v_2y_2 \dots x_mv_my_m$, respectively, where x_i, y_i are moves in A, C , u_i is a sequence of moves in D and v_i is a sequence of moves in B , for each i . So we may define $t \in (\rho|\sigma)|\tau$ to be the interleaving $x_1v_1y_1u_1x_2v_2y_2u_2 \dots x_mv_my_mu_m$.

The converse is immediate: if $t \in (\rho|\sigma)|\tau$ then $t\upharpoonright A, D \in (\rho; \sigma); \tau$. \square

Note that $s \in (\rho|\sigma)|\tau$ if and only if $s\upharpoonright A, B \in \rho$, $s\upharpoonright B, C \in \sigma$, $s\upharpoonright C, D \in \tau$ and $s\upharpoonright A, B, C \in I_{A,B,C}$ and $s\upharpoonright A, C, D \in I_{A,C,D}$. For ν -strategies (pace ordinary strategies on HO-arenas) it is not the case that $(\rho|\sigma)|\tau = \rho|(\sigma|\tau)$ in general, because $s \in (\rho|\sigma)|\tau$ does not entail that $s\upharpoonright B, C, D$ or $s\upharpoonright A, B, D$ are interaction sequences — there may be names introduced in the “hidden components B and C which violate the freshness assumptions. So to prove associativity we must use the saturation of strategies with respect to \sim and the fact that every move has finite support to show that we can always make choices of fresh names which avoid this problem. Specifically, we show for any $s \in (\rho; \sigma); \tau$ there exists $t \in \rho; (\sigma; \tau)$ such that $s\upharpoonright A, D = t\upharpoonright A, D$.

Lemma 3.16 *Given $s \in J_{((A \rightarrow B) \rightarrow C) \rightarrow D}$, suppose $s\upharpoonright A, B, C \in I_{A,B,C}$ and $s\upharpoonright A, C, D \in I_{A,C,D}$, and either $n \in P_\nu(s\upharpoonright B \rightarrow C) \cap P_\nu(s\upharpoonright C \rightarrow D)$ or $n \in (P_\nu(s\upharpoonright B \rightarrow C) \cup P_\nu(s\upharpoonright C \rightarrow D)) \cap O_\nu(s\upharpoonright B \rightarrow D)$. Then $n \notin \nu(s\upharpoonright A, C)$.*

PROOF: Suppose (for a contradiction) that $n \in \nu(s\upharpoonright A, C)$. There are two cases to consider: either $n \in O_\nu(s\upharpoonright A \rightarrow C)$ or $n \in P_\nu(s\upharpoonright A \rightarrow C)$.

If $n \in O_\nu(s\upharpoonright A \rightarrow C)$ then $n \notin P_\nu(s\upharpoonright B \rightarrow C)$ (by the assumption that $s\upharpoonright A, B, C$ is an interaction sequence and therefore satisfies $P_\nu(s\upharpoonright B \rightarrow C) \cap O_\nu(s\upharpoonright A \rightarrow C) = \emptyset$). So $n \in P_\nu(s\upharpoonright C \rightarrow D) \cap O_\nu(s\upharpoonright B \rightarrow D)$, and there is an earliest move b in s such that $b \notin M_C$ and $n \in \nu(b)$. Then one of the following cases must hold:

- b is a P -move in $A \rightarrow B$ — but then $n \in P_\nu(s\upharpoonright A \rightarrow B)$, and so $n \in P_\nu(s\upharpoonright A \rightarrow B) \cap O_\nu(s\upharpoonright A \rightarrow C)$ contradicting the assumption that $s\upharpoonright A, B, C \in I_{A,B,C}$.
- b is an O -move in $A \rightarrow D$ — but then $n \in O_\nu(s\upharpoonright A \rightarrow D) \cap P_\nu(s\upharpoonright C \rightarrow D)$, contradicting $s \in I_{A,C,D}$.
- b is a P -move in $B \rightarrow D$ — but then $n \in P_\nu(s\upharpoonright B \rightarrow D)$, contradicting the assumption that $n \in O_\nu(B \rightarrow D)$.

If $n \in P_\nu(s\upharpoonright A \rightarrow C)$ then $n \notin P_\nu(s\upharpoonright C \rightarrow D)$ (by the assumption that $s\upharpoonright A, C, D \in I_{A,C,D}$) and so $n \in P_\nu(s\upharpoonright B \rightarrow C) \cap O_\nu(s\upharpoonright B \rightarrow D)$. Let b be the first move in s such that $b \notin M_C$ and $n \in \nu(b)$. Then one of the following cases must hold:

- b is a P -move in $A \rightarrow B$: but then $n \in P_\nu(s\upharpoonright A \rightarrow B) \cap P_\nu(s\upharpoonright B \rightarrow C)$, contradicting $s \in I_{A,B,C}$.

- b is an O -move in $A \rightarrow D$ — but then $n \in O_\nu(s \upharpoonright A \rightarrow D) \cap P_\nu(s \upharpoonright A \rightarrow C)$, contradicting $s \in I_{A,C,D}$.
- b is a P -move in $B \rightarrow D$ — but then $n \in P_\nu(s \upharpoonright B \rightarrow D)$, contradicting $n \in O_\nu(s \upharpoonright B \rightarrow D)$.

Hence $n \notin \nu(s \upharpoonright A, C)$. \square

Lemma 3.17 *For any $s \in (\rho|\sigma)|\tau$ there exists:*

$s' \in (\rho|\sigma)|\tau$ such that $s' \upharpoonright B, C, D \in I_{B,C,D}$ and $s' \upharpoonright A, C, D = s \upharpoonright A, C, D$.

$s'' \in (\rho|\sigma)|\tau$ such that $s'' \upharpoonright A, B, D \in I_{A,B,D}$ and $s'' \upharpoonright A, B, D = s \upharpoonright A, B, D$.

PROOF: Similar for both parts: we give the first.

We first note that if $s \upharpoonright A \rightarrow B$, $s \upharpoonright B \rightarrow C$, $s \upharpoonright A \rightarrow C$, $s \upharpoonright C \rightarrow D$ and $s \upharpoonright A \rightarrow D$ satisfy the visibility and well-bracketing conditions, then so does $s \upharpoonright B, D$, as shown in [20,14] etc.

Enumerating $\nu(s) - \nu(s \upharpoonright A, C)$ (in order) as n_1, \dots, n_k , let $f : \mathbb{N} \rightarrow \mathbb{N}$ be the finite partial injection defined:

$$f(n_i) = \min\{\mathbb{N} - (\nu(s) \cup \{f(n_1), \dots, f(n_{i-1})\})\}, 1 \leq i \leq k,$$

$$f(m) = m, m \in \nu(s \upharpoonright A, C).$$

Let $\pi = f$, and define s' by applying π only to moves in s which are from B — i.e. $s' = \pi \upharpoonright_B(s)$, where: $\pi \upharpoonright_B(\varepsilon) = \varepsilon$ and

$$\pi \upharpoonright_B(sa) = \pi \upharpoonright_B(s)(\pi \cdot a) \text{ if } a \in M_B,$$

$$\pi \upharpoonright_B(sa) = \pi \upharpoonright_B(s)a, \text{ otherwise.}$$

By definition, $s' \upharpoonright A, C, D = s \upharpoonright A, C, D$, and $\pi \cdot a = a$ for all $a \in A, C$, and so $s' \upharpoonright A, B = \pi \cdot (s \upharpoonright A, B) \in \rho$, $s' \upharpoonright B, C = \pi \cdot (s \upharpoonright B, C) \in \sigma$ and $s' \upharpoonright C, D = s \upharpoonright C, D \in \tau$, whilst $s' \upharpoonright A, B, C = \pi \cdot (s \upharpoonright A, B, C) \in I_{A,B,C}$ and $s' \upharpoonright A, C, D = s \upharpoonright A, C, D \in I_{A,C,D}$. Thus $s \in (\rho|\sigma)|\tau$.

Finally, observe that by definition of s' , $n \notin \nu(s' \upharpoonright A, C) = \nu(s \upharpoonright A, C)$ implies $n \notin \nu(s' \upharpoonright B) \cap \nu(s' \upharpoonright D)$. So to show that $s \upharpoonright B, C, D \in I_{B,C,D}$, suppose $n \in P_\nu(s' \upharpoonright B \rightarrow C) \cap P_\nu(s' \upharpoonright C \rightarrow D)$ or $n \in ((P_\nu(s' \upharpoonright B \rightarrow C) \cup P_\nu(s' \upharpoonright C \rightarrow D)) \cap O_\nu(s' \upharpoonright B \rightarrow D))$. By Lemma 3.16, $n \notin \nu(s' \upharpoonright A, C)$, and so $n \in (P_\nu(s' \upharpoonright B) \cap O_\nu(s' \upharpoonright D)) \cup ((P_\nu(s' \upharpoonright B) \cup O_\nu(s' \upharpoonright D)) \cap O_\nu(s' \upharpoonright B \rightarrow D))$ — in particular, $n \in \nu(s' \upharpoonright B) \cap \nu(s \upharpoonright D)$. Hence, by contradiction, $P_\nu(s' \upharpoonright B \rightarrow C) \cap P_\nu(s' \upharpoonright C \rightarrow D) = \emptyset$ and $(P_\nu(s' \upharpoonright B \rightarrow C) \cup P_\nu(s' \upharpoonright C \rightarrow D)) \cap O_\nu(s' \upharpoonright B \rightarrow D) = \emptyset$ and so $s \upharpoonright B, C, D \in I_{B,C,D}$ as required. \square

Lemma 3.18 *If $s \in (\rho|\sigma)|\tau$ then there exists $s' \in \rho|(\sigma|\tau)$ such that $s' \upharpoonright A, D = s \upharpoonright A, D$.*

PROOF: Given $s \in (\rho|\sigma)|\tau$, we first apply Lemma 3.17 (second part) to obtain $s'' \in (\rho|\sigma)|\tau$ such that $s'' \upharpoonright A, B, D \in I_{A,B,D}$ and $s'' \upharpoonright A, D = s' \upharpoonright A, D = s \upharpoonright A, D$.

Applying Lemma 3.17 (first part) to s'' , we obtain $s' \in (\rho|\sigma)|\tau$ such that $s'\upharpoonright B, C, D \in I_{B,C,D}$ and $s'\upharpoonright A, D = s''\upharpoonright A, D = s\upharpoonright A, D$. Moreover, $s'\upharpoonright A, C, D = s''\upharpoonright A, C, D$ by construction, and so $s'\upharpoonright A, C, D \in I_{A,C,D}$, and hence $s' \in \rho|(\sigma|\tau)$ as required. \square

Proposition 3.19 *Composition of ν -strategies is associative.*

PROOF: We show that $(\rho; \sigma); \tau \subseteq \rho; (\sigma; \tau)$: proof of the reverse inclusion follows on the same lines. So suppose $s \in (\rho; \sigma); \tau$, then by Lemma 3.15 there exists $t \in (\rho|\sigma)|\tau$ such that $s = t\upharpoonright A, D$. Hence by Lemma 3.18 there exists $t' \in \rho|(\sigma|\tau)$ such that $t'\upharpoonright A, D = t\upharpoonright A, D = s$, and so $s \in \rho; (\sigma; \tau)$ as required. \square

We adopt the standard notion of identity strategy: for any arena A , we define $\text{id}_A : A \rightarrow A = \{s \in L_A \mid \forall t \sqsubseteq^E s. t\upharpoonright A^+ = t\upharpoonright A^-\}$. (where $s \sqsubseteq^E t$ if s is an even-length prefix of t).

Since this strategy (like any other “copycat”) never introduces new names, it must satisfy \sim -saturation and determinacy. Hence we may define a category $\nu\mathcal{G}$ in which the objects are A -rooted ν -arenas, and morphisms from A to B are ν -strategies on $A \rightarrow B$. This has finite coproducts given by the “disjoint union” of arenas: we define $A + B = (M_A + M_B, [\lambda_A, \lambda_B], \vdash_A + \vdash_B, \cdot_A + \cdot_B)$.

We now define *premonoidal* structure on $\nu\mathcal{G}$ which is essentially the same as that described in [11,19], but with restrictions on the sharing of names.

Definition 3.20 *From A -rooted arenas A_1, A_2 , we form the A -rooted arena $A_1 \odot A_2$:*

- $M_{A_1 \odot A_2} = (M_{A_1} \times M_{A_2}^I) \cup (M_{A_1}^I \times M_{A_2})$
- $M_{A_1 \odot A_2}^I = M_{A_1}^I \times M_{A_2}^I$,
- $\lambda_{A_1 \odot A_2}(\langle m_1, m_2 \rangle) = \lambda_{A_2}(m_2)$, if $m_1 \in M_{A_1}^I$,
 $\lambda_{A_1 \odot A_2}(\langle m_1, m_2 \rangle) = \lambda_{A_1}(m_1)$, otherwise,
- $\vdash_{A_1 \odot A_2} = (\vdash_{A_1} \otimes \text{Id}) \cup (\text{Id} \otimes \vdash_{A_2})$.
- $\pi \cdot \langle m, n \rangle = \langle \pi \cdot m, \pi \cdot n \rangle$,

The one-move arena I is the unit for \odot .

Given a legal sequence $t \in L_{A_1 \odot A_2}$, we obtain a legal sequence $t\upharpoonright A_i \in L_{A_i}$ by taking the i th projection from each move, and then erasing all initial moves except the first from the result. For each ν -arena A we define an endofunctor $-\odot A : \nu\mathcal{G} \rightarrow \nu\mathcal{G}$. Given $\sigma : B \rightarrow C$, let $\sigma \odot A : B \odot A \rightarrow C \odot A =$

$$\{s \in L_{B \odot A \rightarrow C \odot A} \mid s\upharpoonright B, C \in \sigma \wedge s\upharpoonright A, A \in (\text{id}_A \cup M_A^I) \wedge P_\nu(s\upharpoonright B \rightarrow C) = P_\nu(s)\}.$$

Lemma 3.21 *$\sigma \odot A$ is a well-defined strategy.*

PROOF: The key property to establish is \sim -determinacy, which we prove using

Lemma 3.9 as for preservation of \sim -determinacy by composition. Suppose $saa', tbb' \in \sigma \odot A$ and there exists $\pi \in G$ such that $\pi \cdot (sa) = tb$. Then if a' is a move in (either occurrence of A) it is a copy of a' , and so $\pi \cdot a = b$ implies $\pi \cdot a' = b'$, and $saa' \sim tbb'$ as required. If a' is a move in B, C then $sa \upharpoonright B, C \sim tb \upharpoonright B, C$ and so by \sim -determinacy of σ , there exists $\pi' \in G$ such that $\pi' \cdot (saa' \upharpoonright B, C) = tbb'$. By the condition $P_\nu(saa' \upharpoonright B \rightarrow C) = P_\nu(saa')$, if $n \in \nu(sa) \cap \nu(a')$ then $n \in \nu(sa \upharpoonright B, C)$ and so $\pi(n) = \pi'(n)$, and if $m \in \nu(sa)$ and $n \in \nu(saa') - \nu(sa)$ then $\pi(m) \in \nu(sa)$ and $\pi'(m) \in \nu(tbb') - \nu(tb)$ and so $\pi'(m) \neq \pi'(n)$ and so by Lemma 3.9 $saa' \sim tbb'$ as required. \square

Proposition 3.22 $(\nu\mathcal{G}, I, \odot)$ is a symmetric premonoidal category.

PROOF: This follows [11,19]. \square

We now identify, via conditions on strategies, a category of arenas and ν -strategies for which the premonoidal product is cartesian. The first condition is *totality*, as in [11].

Definition 3.23 A morphism $f : A \rightarrow B$ is total if $g; f = \perp$ implies $g = \perp$ (where \perp is the empty strategy). So $\sigma : A \rightarrow B$ is total if it responds to each initial question in A with an answer in B . A sequence in which this occurs is said to be total.

Definition 3.24 A total sequence qas is single-threaded if:

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

A total strategy σ is single-threaded if every non-empty sequence in σ is single-threaded.

To define the composition of single-threaded strategies we apply a “promotion” operation $(-)^{\dagger}$ converting a single-threaded strategy σ to a *thread independent* strategy σ^{\dagger} . This essentially follows the definitions of e.g. [5,10], except that we require additional conditions preventing the sharing of names across threads.

Definition 3.25 The thread of a total sequence $qasb \in L_{A \rightarrow B}$ is a legal sequence of $A \rightarrow B$ defined (following e.g. [5]) as follows:

$\text{thread}(qasb) = qab$ if b is justified by a ,

$\text{thread}(qasb) = \text{thread}(qas)b$ if b is a move in A .

$\text{thread}(qasbtc) = \text{thread}(qasb)c$ if b and c are moves in B and b is the last move in $qasbt$ with the same hereditary justifier as c .

For completeness, we let $\text{thread}(s) = s$ for $s \sqsubseteq qa$.

Note that $\mathbf{thread}(gas)$ is a legal sequence by the visibility condition. In particular, if b is a Player move, then $\mathbf{thread}(gasb) = \mathbf{thread}(gas)b$.

Definition 3.26 A total (legal) sequence gas on $A \rightarrow B$ is *thread-independent* with respect to naming if any name in the support of a Player move which is fresh in its thread is fresh in the whole sequence gas — i.e. if $qatb \sqsubseteq^E gas$ then $(\nu(\mathbf{thread}(qatb)) - \nu(\mathbf{thread}(qat))) \cap \nu(qat) = \emptyset$. We write $\mathbf{tot}(A, B)$ for the set of total and thread-independent sequences on $A \rightarrow B$, and define $\sigma^\dagger = \{\varepsilon\} \cup \{s \in \mathbf{tot}(A, B) \mid \forall t \sqsubseteq^E s. \mathbf{thread}(t) \in \sigma\}$.

Lemma 3.27 σ^\dagger is a well-defined strategy.

PROOF: To show \sim -determinacy, suppose $sa, tb \in \sigma^\dagger$ and $s \sim t$. Then $\mathbf{thread}(s) \sim \mathbf{thread}(t)$ and hence $\mathbf{thread}(sa) \sim \mathbf{thread}(tb)$. So we have automorphisms $\pi, \pi' : \mathbb{N} \rightarrow \mathbb{N}$ such that $\pi \cdot s = t$ and $\pi' \cdot \mathbf{thread}(sa) = \mathbf{thread}(tb)$, and so in particular $\pi' \cdot a = b$. We verify that the further conditions of Lemma 3.9 are satisfied. If $n \in \nu(s) \cap \nu(a)$ — i.e. n is not fresh in sa — then by the thread independence condition for names, n is not fresh in $\mathbf{thread}(sa)$ — i.e. $n \in \nu(\mathbf{thread}(sa))$ and hence $\pi'(n) = \pi(n)$. If $m \in \nu(s)$ and $n \in \nu(a)$ and $\pi(m) = \pi'(n)$, then $\pi'(n)$ is not fresh in tb , and hence it is not fresh in $\mathbf{thread}(tb)$ — i.e. $n \in \nu(\mathbf{thread}(s)) \subseteq \nu(s)$. So by Lemma 3.9, $sa \sim tb$ as required. \square

We define a category $\nu\mathcal{G}_t$ with A -rooted arenas as objects and single-threaded total strategies on $A \rightarrow B$ as morphisms from A to B . Composition of $\sigma : A \rightarrow B$ and $\tau : B \rightarrow C$ is defined $\sigma^\dagger; \tau$, and the identity \mathbf{der}_A is the subset of \mathbf{id}_A consisting of single-threaded sequences. To prove that $\nu\mathcal{G}_t$ is a category, we establish the following properties for $(-)^{\dagger}$ (following e.g. [20]).

Lemma 3.28 For any σ, τ :

- $\mathbf{der}_A^\dagger = \mathbf{id}_A$,
- $\sigma^\dagger; \mathbf{der}_A = \sigma$,
- $\sigma^\dagger; \tau^\dagger = (\sigma^\dagger; \tau)^\dagger$.

Proposition 3.29 The category $\nu\mathcal{G}_t$ has cartesian products (given by \odot).

PROOF: We have copycat (single-threaded) strategies yielding natural maps $\pi_i : A_1 \odot A_2 \rightarrow A_i$ and for single-threaded strategies $\sigma : A \rightarrow B$ and $\tau : A \rightarrow C$ we may define the pairing $\langle \sigma, \tau \rangle$ to consist of the total and single-threaded sequences s such that $s \upharpoonright A, B \in \sigma$ and $s \upharpoonright A, C \in \tau$. \square

By Lemma 3.28, $(-)^{\dagger}$ is a functor from $\nu\mathcal{G}_t$ to $\nu\mathcal{G}$. We may also check that it is a premonoidal functor: for any single-threaded strategy $\sigma : A \rightarrow B$, $(\sigma \times \mathbf{der}_C)^\dagger = \sigma^\dagger \odot C$. Thus we have a Freyd category [26]; a cartesian category $\nu\mathcal{G}_t$, a symmetric premonoidal category $\nu\mathcal{G}$, and an identity-on-objects strict symmetric premonoidal functor $(-)^{\dagger} : \nu\mathcal{G}_t \rightarrow \nu\mathcal{G}$. Moreover, it is a *closed* Freyd

category: the functor $(-)^{\dagger} \odot A : \nu\mathcal{G}_t \rightarrow \nu\mathcal{G}$ has a right adjoint $A \multimap - : \nu\mathcal{G} \rightarrow \nu\mathcal{G}_t = \uparrow(A \multimap -)$.

Proposition 3.30 $A \multimap -$ is right adjoint to $(-)^{\dagger} \odot A : \nu\mathcal{G}_t \rightarrow \nu\mathcal{G}$.

PROOF: There is an obvious bijection from legal sequences on $A \odot B \rightarrow C$ to single-threaded sequences on $A \rightarrow (B \multimap C)$, sending $\langle m, n \rangle s$ to *mans*. Thus for each morphism $\sigma : A \odot B \rightarrow C$, we define a unique single-threaded strategy $\Lambda(\sigma) : A \rightarrow (B \multimap C) = \{\varepsilon\} \cup \{ma \mid m \in M_A^I\} \cup \{mans \mid \langle m, n \rangle s \in \sigma\}$ such that $(\Lambda(\sigma))^{\dagger} \odot B; \mathbf{app}_{B,C} = \sigma$, where the co-unit $\mathbf{app}_{B,C} : (B \multimap C) \odot B \rightarrow C$ is derived from $\mathbf{id}_{B \multimap C}$ by the above bijection. \square

Thus we may interpret the call-by-value function type $S \Rightarrow T$ as $\llbracket S \rrbracket \multimap \llbracket T \rrbracket$, and terms-in-context $x_1 : S_1, \dots, x_n : S_n \vdash M : T$ of the λ -calculus as morphisms from $\llbracket S_1 \rrbracket \odot \dots \odot \llbracket S_n \rrbracket$ to $\llbracket T \rrbracket$ in $\nu\mathcal{G}$:

$$\begin{aligned} \llbracket x_1 : S_1, \dots, x_n : S_n \vdash x_i : S_i \rrbracket &= \pi_i \\ \llbracket \Gamma \vdash M N : T \rrbracket &= \delta_{\Gamma}^{\dagger}; (\llbracket \Gamma \vdash M : S \Rightarrow T \rrbracket \odot \llbracket \Gamma \vdash N : S \rrbracket); \mathbf{app}_{\llbracket S \rrbracket, \llbracket T \rrbracket} \\ \llbracket \Gamma \vdash \lambda x. M : S \Rightarrow T \rrbracket &= \Lambda(\llbracket \Gamma, x : S \vdash M : T \rrbracket)^{\dagger}. \end{aligned}$$

3.3 Semantics of the nu-calculus

We may interpret the nu-calculus in $\nu\mathcal{G}$ by fixing the interpretation of the ground types to be $\llbracket o \rrbracket = I + I$ — from which we derive the interpretation of the truth values and conditional — and $\llbracket \nu \rrbracket = N$, with the remaining constants for new name generation and equality testing being interpreted as strategies which we have already described i.e. $\llbracket \mathbf{new} \rrbracket = \mathbf{new}$ and $\llbracket \mathbf{eq} \rrbracket = \Lambda(\mathbf{eq})$. To verify that this yields a sound model of the nu-calculus, we may observe that it satisfies the requirements given by Stark in [29] for a categorical model of the nu-calculus. These are based on monadic models of the computational λ -calculus, but transfer readily to the more direct description used here. (Note that $\nu\mathcal{G}$ is equivalent to the Kleisli category of the strong monad $\mathbf{T}A = I \multimap A$ on $\nu\mathcal{G}_t$.) Thus we have the categorical structure and properties required in [29] to interpret the nu-calculus, which may be summarised as follows:

- A semantics of the call-by-value λ -calculus — the closed Freyd category $(\nu\mathcal{G}, \nu\mathcal{G}_t, \odot, I)$.
- To interpret the type o of booleans, a disjoint coproduct of the terminal object (of $\nu\mathcal{G}_t$) with itself: as we have noted, $\nu\mathcal{G}$ has all (disjoint) coproducts.
- To interpret the type ν of names, a distinguished object N which is decidable — i.e. the map $\mathbf{eq}' : N \odot N \rightarrow I + I = \Lambda^{-1}(\mathbf{eq})$ completes the pullback square $t_N; \mathbf{in}_l : N \rightarrow I + I = \langle \mathbf{id}, \mathbf{id} \rangle^{\dagger}; \mathbf{eq}$.
- To interpret the new-name declaration $\mathbf{new} : \nu$, a distinguished map $\mathbf{new} : I \rightarrow N$, satisfying three further equations given in a computational meta-language in [29]. These stipulate that new names are distinct from all others,

the order in which they are generated is not relevant, and that unused names may be ignored.

Proposition 3.31 $\nu\mathcal{G}$ yields a categorical model of the nu-calculus in the sense of [29].

(We omit further details, since establishing soundness for the model of $\lambda\nu!$ provides an alternative proof). As examples we consider two contextual equivalences of the nu-calculus described in [28]. The first, $\nu x.\lambda y.x = y \approx_{\nu \Rightarrow o} \lambda y.\mathbf{ff}$ holds in our model: the term $\nu x.\lambda y.x = y : \nu \Rightarrow o$ is interpreted by composing $\mathbf{new} : I \rightarrow N$ with $\llbracket x : \nu \vdash \lambda y.x = y \rrbracket : N \Rightarrow (N \rightarrow I + I) = \mathbf{eq}$. As we have already observed, this is equal to $\{gam_i\mathbf{ff} \mid i \in \mathbb{N}\} = \llbracket \lambda y.\mathbf{ff} \rrbracket$. The second is the equivalence $\lambda f : \nu \Rightarrow o.\mathbf{tt} \approx \nu x.\nu y.\lambda f : \nu \Rightarrow o.(f x = f y)$, which, as we have observed holds in the nu-calculus, but not in $\lambda\nu!$. These terms are not denotationally equivalent in our model either. In the former, Player supplies the value \mathbf{tt} immediately, whereas in the latter, Player queries the argument $\nu \Rightarrow o$ twice, supplying it with distinct names on each occasion (see Figure 1).

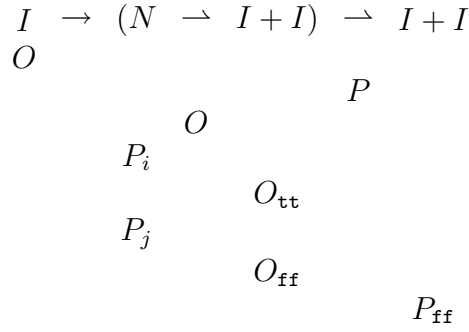


Fig. 1. A typical play in $\llbracket \nu x.\nu y.\lambda f.(f x = f y) \rrbracket$.

4 Semantics of Pointers

Although the latter example shows that our semantics of the nu-calculus is not fully abstract, it does not preclude the possibility of using $\nu\mathcal{G}$ to construct a fully abstract model of $\lambda\nu!$. Our method of doing this is quite simple and direct; we add a store — in the form of a finite subset of $\mathbb{N} \times \mathbb{N}$ — to each move, and allow the group G to act on these moves-with-store. This associates interactions with state change

Definition 4.1 Let A be a ν -arena. A move-with-store over A is a pair $\langle m, \mathcal{S} \rangle$, where $m \in M_A$, and $\mathcal{S} \subseteq_{fin} \mathbb{N} \times \mathbb{N}$ is a store — i.e. $(i, j), (i, k) \in \mathcal{S}$ implies $j = k$. G acts pointwise on moves-with-store: $\pi \cdot \langle m, \mathcal{S} \rangle = \langle \pi \cdot m, \{(\pi(i), \pi(j)) \mid (i, j) \in \mathcal{S}\} \rangle$.

A *justified sequence-with-store* s over a ν -arena A is a sequence of moves-with-store $\langle a_1, \mathcal{S}_1 \rangle \dots \langle a_n, \mathcal{S}_n \rangle$ with justification pointers, such that $\text{proj}(s) = a_1 \dots a_n$ is a justified sequence on A and $\text{dom}(\mathcal{S}_i) \subseteq \text{dom}(\mathcal{S}_{i+1})$ for all $i < n$.

It is quite straightforward to give a semantics of $\lambda\nu!$ using sequences-with-store. For example, we have the following strategies for assignment and dereferencing: the first takes two names (and a store) and returns tt with the store updated with the appropriate assignment. The second takes a name n and a store, and returns the value stored at n , together with the unchanged store.

- $\text{ass} : N \odot N \rightarrow I + I = \{\varepsilon\} \cup \{\langle \langle i, j \rangle, \mathcal{S} \rangle \langle \text{tt}, \mathcal{S}[i \mapsto j] \rangle\} \mid i, j \in \mathbb{N}, \mathcal{S} \in \mathbb{N} \rightarrow \mathbb{N}\}$
- $\text{deref} : N \rightarrow N = \{\varepsilon\} \cup \{\langle i, \mathcal{S} \rangle \langle j, \mathcal{S} \rangle \mid i \in \mathbb{N} \wedge (i, j) \in \mathcal{S}\}$.

Using the constructions described in the previous section we may construct a model of the nu-calculus, and use the strategies ass and deref to interpret the constants ass and deref . However, this semantics is not fully abstract for a simple reason: strategies will typically contain references to pointers which are no longer accessible, and cannot therefore be the basis of a distinction by an observing context. For example, the term $\nu x. \text{tt} : o$ will be interpreted as the strategy on $I \rightarrow N$ which responds to the initial question $\langle q, \mathcal{S} \rangle$ with $\langle \text{tt}, \mathcal{S} \cup \{(i, i)\} \rangle$ for some $i \notin \text{dom}(\mathcal{S})$ — i.e. creating a new name and adding a reflexive pointer to it to the store. But since the declaration of x does not affect any observable part of the store, $\nu x. \text{tt}$ is contextually equivalent to tt in $\lambda\nu!$. To cut down our model, and obtain a full abstraction result, we remove (“garbage-collect”) all such inaccessible pointers.

Definition 4.2 We define the set of revealed or global names of sequences with store to be the least set such that:

$\text{glob}(s), \nu(a) \subseteq \text{glob}(s\langle a, \mathcal{S} \rangle)$ and if $i \in \text{glob}(s\langle a, \mathcal{S} \rangle)$ and $(i, j) \in \mathcal{S}$, then $j \in \text{glob}(s\langle a, \mathcal{S} \rangle)$.

A sequence-with-store $t = \langle a_1, \mathcal{S}_1 \rangle \dots \langle a_n, \mathcal{S}_n \rangle$ over an arena A is legal ($t \in L_A^S$) if $\text{proj}(t) \in L_A$ and t satisfies the locality condition: all and only the locations with revealed names may appear in the store — i.e. if $s\langle a, \mathcal{S} \rangle \sqsubseteq t$ then $i \in \text{dom}(\mathcal{S})$ if and only if $i \in \text{glob}(s\langle a, \mathcal{S} \rangle)$.

A strategy-with-store for the arena A is an even-prefix-closed subset of L_A^S which satisfies the \sim -saturation and \sim -determinacy conditions.

We now extend composition to strategies-with-store. If we form the restriction of a (legal) sequence-with-store to some sub-arena simply by removing moves, then the result will not, in general, satisfy the locality condition since some of the global names may become hidden. Instead, we define restriction so that inaccessible pointers are removed. We define the garbage-collection operation $\gamma(_)$ on sequences-with-store as follows:

$\gamma(\varepsilon) = \varepsilon$ and $\gamma(s\langle a, \mathcal{S} \rangle) = \gamma(s)\langle a, \{\langle i, j \rangle \in \mathcal{S} \mid i \in \text{glob}(s\langle a, \mathcal{S} \rangle)\} \rangle$.

Definition 4.3 *The set of interaction sequences-with-store $I_{A,B,C}^S$ consists of the justified sequences-with-store t over $(A \rightarrow B) \rightarrow C$ satisfying the freshness conditions of Definition 3.7 together with the following condition (which prevents participants in the interaction from changing parts of the store which are not accessible to them). If $s = s'\langle a, \mathcal{S} \rangle \langle b, \mathcal{S}' \rangle \sqsubseteq t$ then:*

- *if b is a Player move in $A \rightarrow B$ then for all $n \in \nu(s) - \nu(\gamma(s \upharpoonright A, B))$, $\mathcal{S}(n) = \mathcal{S}'(n)$.*
- *if b is a Player move in $B \rightarrow C$ then for all $n \in \nu(s) - \nu(\gamma(s \upharpoonright B, C))$, $\mathcal{S}(n) = \mathcal{S}'(n)$.*
- *if b is an O-move in $A \rightarrow C$ then for all $n \in \nu(s) - \nu(\gamma(s \upharpoonright A, C))$, $\mathcal{S}(n) = \mathcal{S}'(n)$.*

We may thus define the “parallel composition plus hiding and garbage collection” of strategies-with-store $\sigma : A \rightarrow B$ and $\tau : B \rightarrow C$ as for regular ν -strategies: $\sigma \upharpoonright \tau = \{s \in I_{A,B,C}^S \mid \gamma(s \upharpoonright A, B) \in \sigma \wedge \gamma(s \upharpoonright B, C) \in \tau\}$ and $\sigma; \tau = \{s \in L_{A \rightarrow B}^S \mid \exists t \in \sigma \upharpoonright \tau. \gamma(t \upharpoonright A, C) = s\}$.

Lemma 4.4 *The composition of strategies-with-store is well-defined and associative.*

PROOF: The proof that $\sigma; \tau$ satisfies \sim -saturation and \sim -determinacy follows the same lines as the proof of Proposition 3.12. We now require the additional condition on interaction sequences to prove Lemma 3.10: Given $s \langle a, \mathcal{S}_1 \rangle \langle b, \mathcal{S}_2 \rangle, s' \langle a', \mathcal{S}'_1 \rangle \langle b', \mathcal{S}'_2 \rangle$ such that (e.g.) b is a Player move in $A \rightarrow B$, suppose we have automorphisms $\pi, \pi' \in G$ such that $\pi \cdot (s \langle a, \mathcal{S}_1 \rangle) = s' \langle a', \mathcal{S}'_1 \rangle$ and $\pi' \cdot \gamma(s \langle a, \mathcal{S}_1 \rangle \langle b, \mathcal{S}_2 \rangle \upharpoonright A, C) = \gamma(s' \langle a', \mathcal{S}'_1 \rangle \langle b', \mathcal{S}'_2 \rangle \upharpoonright A, C)$. Then we construct an automorphism $\pi'' \in G$ such that $\pi'' \cdot (s \langle a, \mathcal{S}_1 \rangle \langle b, \mathcal{S}_2 \rangle) = s' \langle a', \mathcal{S}'_1 \rangle \langle b', \mathcal{S}'_2 \rangle$ as in Lemma 3.9, using the freshness assumptions on interaction sequences as in the original proof of Lemma 3.10, plus the fact that if $n \in \nu(s \langle a, \mathcal{S}_1 \rangle) - \nu(\gamma(s \langle a, \mathcal{S}_1 \rangle \langle b, \mathcal{S}_2 \rangle \upharpoonright A, C))$, then $\pi \cdot \mathcal{S}_2(n) = \pi \cdot \mathcal{S}_1(n) = \mathcal{S}'_1(\pi(n)) = \mathcal{S}'_2(\pi(n))$.

To establish associativity of composition, we extend the proof of Proposition 3.19 in similar fashion. \square

Thus we may define the category $\nu\mathcal{G}^s$ in which objects are A -rooted ν -arenas and morphisms from A to B are strategies-with-store on $A \rightarrow B$. We define a closed Freyd category based on $\nu\mathcal{G}^s$, using essentially the same constructions as for $\nu\mathcal{G}$, except that, as in the case of composition, we require additional conditions stipulating that a part of the store which is not accessible cannot be changed. For instance, to extend the operation $(_)^\dagger$ to strategies-with-store we define the notion of total and thread-independent sequence-with-store: A total sequence-with-store qas on $A \rightarrow B$ is thread-independent ($qas \in \text{tot}^S(A, B)$) if it is thread-independent with respect to names (i.e. $\forall tb \sqsubseteq^E s. (\nu(\gamma(\text{thread}(tb))) - \nu(\gamma(\text{thread}(t)))) \cap \nu(t) = \emptyset$) and for all $t \langle a, \mathcal{S} \rangle \langle b, \mathcal{S}' \rangle \sqsubseteq^E s$ and $n \in \nu(t \langle a, \mathcal{S} \rangle) - \nu(\gamma(\text{thread}(t \langle a, \mathcal{S} \rangle))$, $\mathcal{S}(n) = \mathcal{S}'(n)$.

Definition 4.5 We define the category $\nu\mathcal{G}_t^s$ in which objects are A -rooted ν -arenas and morphisms from A to B are single-threaded total strategies-with-store on $A \rightarrow B$. Composition of $\sigma : A \rightarrow B$ with $\tau : B \rightarrow C$ is defined to be $\sigma^\dagger; \tau$, where $\sigma^\dagger = \{s \in \text{tot}^S(A \rightarrow B) \mid \forall tb \sqsubseteq^E s. \gamma(\text{thread}(tb)) \in \sigma\}$

As in $\nu\mathcal{G}_t$, $\nu\mathcal{G}_t^s$ has finite products given by $_ \odot _$. Thus we may prove the following using the same constructions and arguments as for $\nu\mathcal{G}$ and $\nu\mathcal{G}_t$.

Proposition 4.6 The premonoidal category $(\nu\mathcal{G}^s, \odot, I)$, the cartesian category $\nu\mathcal{G}_t^s$ and the functor $(_)^\dagger$ form a closed Freyd Category.

Thus we may define a structure-preserving (identity-on-objects) embedding $J : \nu\mathcal{G} \rightarrow \nu\mathcal{G}^s$.

Definition 4.7 For each arena A , we define the set of write-independent sequences-with-store (sequences in which Player only changes the store by initializing new variables) $\text{WI}_A = \{s \in L_A^S \mid t\langle a, \mathcal{S} \rangle \langle b, \mathcal{S}' \rangle \sqsubseteq^E s \implies \mathcal{S}' = \mathcal{S} \cup \{(i, i) \mid i \in \nu(b) - \nu(t\langle a, \mathcal{S} \rangle)\}\}$.

We define $J(\sigma : A) = \{s \in \text{WI}_A \mid \exists t \in \sigma.\text{proj}(s) = t\}$.

We may interpret the types and constants of the nu-calculus by J -embedding. Thus we may complete the interpretation of $\lambda\nu!$ simply by using the strategies **ass** and **deref** to interpret the constants **ass** and **deref** — i.e. we define: $\llbracket \text{ass} \rrbracket = \Lambda(\Lambda(\text{ass}))$ and $\llbracket \text{deref} \rrbracket = \Lambda(\text{deref})$. Note that if T is a ν -free type, then every legal sequence-with-store $s \in L_{I \rightarrow [T]}^S$ has no globally accessible names and therefore corresponds to an ordinary legal sequence on $L_{I \rightarrow [T]}$. At these types, our model is equivalent to Abramsky and McCusker's model of RML [4].

To prove soundness of our interpretation, we define the semantic interpretation of the environment (k, \mathcal{S}) as a strategy on $I \rightarrow N^k$ as follows: $\llbracket (k, \mathcal{S}) \rrbracket : I \rightarrow N^k = \{\varepsilon\} \cup \{\langle q, \emptyset \rangle \langle \langle a_0, \dots, a_{k-1} \rangle, \{(a_0, a_{\mathcal{S}(0)}), \dots, (a_{k-1}, a_{\mathcal{S}(k-1)})\} \mid a_i = a_j \implies i = j\}$ i.e. $\llbracket (k, \mathcal{S}) \rrbracket$ generates k new names and performs the assignments specified by \mathcal{S} . We may now define the interpretation of a program-in-environment $\llbracket M : T, (k, \mathcal{S}) \rrbracket : I \rightarrow [T] = \llbracket (k, \mathcal{S}) \rrbracket; \llbracket M \rrbracket$,

Lemma 4.8 The following equations hold:

- $\llbracket (\lambda x.M) \text{new}, (k, \mathcal{S}) \rrbracket = \llbracket (\lambda x.M) k, (k+1, \mathcal{S} \cup \{(k, k)\}) \rrbracket$,
- $\llbracket (\lambda x.M) (i := j), (k, \mathcal{S}) \rrbracket = \llbracket (\lambda x.M) \text{tt}, (k, \mathcal{S}[i \mapsto j]) \rrbracket$,
- $\llbracket (\lambda x.M) !i, (k, \mathcal{S}) \rrbracket = \llbracket (\lambda x.M) j, (k, \mathcal{S}) \rrbracket$ ($\mathcal{S}(i) = j$).

PROOF: By definition of the strategy **ass**, we have:

$$\llbracket k, \mathcal{S} \rrbracket; \delta_{N^k}; (\langle \pi_i, \pi_j \rangle^\dagger; \text{ass}) \odot N^k = \llbracket (k, \mathcal{S}[i \mapsto j]) \rrbracket; (\text{in}_1^\dagger \odot N^k)$$

where $\delta_A : A \rightarrow A \odot A$ is the “premonoid diagonal” $\langle \text{id}_A, \text{id}_A \rangle^\dagger$.

$$\begin{aligned}
& \llbracket (\lambda x.M) (i := j), (k, \mathcal{S}) \rrbracket = \llbracket k, \mathcal{S} \rrbracket; \llbracket (\lambda x.M) (i := j) \rrbracket \\
& = \llbracket k, \mathcal{S} \rrbracket; \delta_{N^k}; (\llbracket i := j \rrbracket \odot N^k); \llbracket x, k \vdash M \rrbracket \\
& = \llbracket k, \mathcal{S} \rrbracket; \delta_{N^k}; ((\langle \pi_i, \pi_j \rangle^\dagger; \text{ass}) \odot N^k); \llbracket x, k \vdash M \rrbracket \\
& = \llbracket (k, \mathcal{S}[i \mapsto j]) \rrbracket; (\llbracket \text{tt} \rrbracket^\dagger \odot N^k); \llbracket x, k \vdash M \rrbracket \\
& = \llbracket \lambda x.M \text{tt}, (k, \mathcal{S}[i \mapsto j]) \rrbracket \text{ as required.}
\end{aligned}$$

Similarly, by definition of the strategy deref , if $\mathcal{S}(i) = j$, then

$$\begin{aligned}
& \llbracket k, \mathcal{S} \rrbracket; \delta_{N^k}; (\pi_i^\dagger; \text{deref}^\dagger) \odot N^k = \llbracket k, \mathcal{S} \rrbracket; \delta_{N^k}; \pi_j^\dagger \odot N^k. \\
& \text{Hence } \llbracket (\lambda x.M) !i, (k, \mathcal{S}) \rrbracket = \llbracket k, \mathcal{S} \rrbracket; \llbracket (\lambda x.M) !i \rrbracket \\
& = \llbracket k, \mathcal{S} \rrbracket; \delta_{N^k}; (\llbracket !i \rrbracket \odot N^k); \llbracket x, k \vdash M \rrbracket \\
& = \llbracket k, \mathcal{S} \rrbracket; \delta_{N^k}; ((\pi_i^\dagger; \text{deref}^\dagger) \odot N^k); \llbracket x, k \vdash M \rrbracket \\
& = \llbracket k, \mathcal{S} \rrbracket; \delta_{N^k}; (\pi_j^\dagger \odot N^k); \llbracket x, k \vdash M \rrbracket = \llbracket \lambda x.M j, (k, \mathcal{S}) \rrbracket. \quad \square
\end{aligned}$$

Lemma 4.9 *If $M, (k, \mathcal{S}) \Downarrow V, (k', \mathcal{S}')$ then for any program U such that $k \vdash U$, $\llbracket U M, (k, \mathcal{S}) \rrbracket = \llbracket U V, (k', \mathcal{S}') \rrbracket$.*

PROOF: By induction on the derivation of $M, (k, \mathcal{S}) \Downarrow V, (k', \mathcal{S}')$, using the above equations and the properties of $\nu\mathcal{G}^s$ as a model of the nu-calculus.

For example, suppose $M \equiv M' := M''$, where $M', (k', \mathcal{S}') \Downarrow i, (k', xs')$ and $M'', (k', \mathcal{S}') \Downarrow j, (k'', \mathcal{S}'')$. Then for any $k \vdash U$, $\llbracket U M, k, \mathcal{S} \rrbracket = \llbracket (\lambda x.U (x := M'')) M', (k, \mathcal{S}) \rrbracket$

$$\begin{aligned}
& = \llbracket (\lambda x.U x := M'') i, (k', \mathcal{S}') \rrbracket = \llbracket U (i := M''), (k', \mathcal{S}') \rrbracket \\
& = \llbracket (\lambda y.U (i := y)) M'', (k', \mathcal{S}') \rrbracket = \llbracket (\lambda y.U (i := y)) j, (k'', \mathcal{S}'') \rrbracket \\
& = \llbracket U (i := j), (k'', xs'') \rrbracket = \llbracket U \text{tt}, (k'', \mathcal{S}''[i \mapsto j]) \rrbracket \text{ as required.} \quad \square
\end{aligned}$$

Theorem 4.10 *For any closed $M : o$, $M \Downarrow \text{tt}$ if and only if $\llbracket M \rrbracket = \llbracket \text{tt} \rrbracket$.*

PROOF: If $M \Downarrow \text{tt}, (k, \mathcal{S})$ then $\llbracket M \rrbracket = \llbracket (\lambda x.x) M \rrbracket = \llbracket (\lambda x.x) \text{tt}, (k, \mathcal{S}) \rrbracket = \llbracket \text{tt} \rrbracket$. If $M \not\Downarrow \text{tt}$ then $M \Downarrow \text{ff}$ and so $\llbracket M \rrbracket = \llbracket \text{ff} \rrbracket$. \square

5 Definability and Full Abstraction

Rather than proving full abstraction for our model of $\lambda\nu!$ directly, we shall establish it for a “partial” extension $\lambda\nu!_\perp$ containing a single additional constant $\Omega : o$ representing a divergent program. The operational semantics of $\lambda\nu!_\perp$ are the same as for $\lambda\nu!$: we write $M \Downarrow$ if there exists V, \mathcal{E} such that $M, (0, \emptyset) \Downarrow V, \mathcal{E}$. Thus we may define observational approximation in $\lambda\nu!_\perp$ — $M \lesssim N$ if for all closing contexts $C[-] : o$, $C[M] \Downarrow$ implies $C[N] \Downarrow$.

We may show that observational equivalence in $\lambda\nu!_\perp$ is conservative over observational equivalence in $\lambda\nu!$, by using a simple translation.

Definition 5.1 For each program M of $\lambda\nu!_{\perp}$, and a variable or name $a : \nu$, let M_a be the program obtained by replacing each occurrence of Ω in M with the term $a := \text{new}$.

Lemma 5.2 Suppose a is a variable not occurring in M . Then $\nu a.(M_a; !a = a) \Downarrow \text{tt}$ if and only if $M \Downarrow$.

PROOF: For any environment $\mathcal{E} = (k, \mathcal{S})$, let $\mathcal{E}' = (k+1, \mathcal{S}')$, where $\mathcal{S}'(0) = 0$, and $\mathcal{S}'(i+1) = \mathcal{S}(i) + 1$, for $1 < i < k$. We show that $M, \mathcal{E}_1 \Downarrow V, \mathcal{E}_2$ if and only if $M_0, \mathcal{E}'_1 \cup \Downarrow V, \mathcal{E}'_2$ by induction on derivation. \square

Proposition 5.3 For any terms $M, N : T$ of $\lambda\nu!$, $M \approx N$ if and only if $M \lesssim N$ and $N \lesssim M$.

PROOF: From left to right, this is straightforward. For the converse, suppose e.g. that $M \not\lesssim N$. Then there is a context $C[_]$ such that $C[M] \Downarrow$ and $C[N] \not\Downarrow$. Let a be a variable not occurring in $C[M]$ or $C[N]$. By Lemma 5.2 $\nu a.(C[M]_a; !a = a) \Downarrow \text{tt}$ and $\nu a.(C[N]_a; !a = a) \not\Downarrow \text{tt}$ and so $M \not\approx N$ as required. \square

We now prove inequational full abstraction of our model of $\lambda\nu!_{\perp}$ with respect to the inclusion order on strategies. (Note that this is complete and algebraic: a compact strategy is one in which the set of orbits $\{[s] \mid s \in \sigma\}$ is finite.) We prove that each compact strategy $\sigma : I \rightarrow \llbracket T \rrbracket$ (where T is a type of $\lambda\nu!$) is definable as a term $M_{\sigma} : T$ of $\lambda\nu!_{\perp}$ by combining two familiar proof techniques; factorization of strategies into the composition of a series of canonical side-effects (new-name creation, assignment and dereferencing) and a functional part, together with a proof by decomposition that the latter is definable as a λ -term with conditionals, divergence, truth values and equality testing.

As a preliminary, we observe that for any legal sequence-with-store s over a $\lambda\nu!$ type-object (in which each move has either singleton or empty support) we may define a total ordering on $\nu(s)$. Essentially, this is the order in which they are introduced in s : moves introduced in the same store-move are ordered according to the order of their locations.

Definition 5.4 For each legal sequence-with-store over a $\lambda\nu!$ -arena we define the order \ll_s on $\nu(s)$ to be the smallest subset of $\nu(s) \times \nu(s)$ such that:

- if $\exists t \sqsubseteq s$ such that $n \in \nu(t)$ and $m \notin \nu(t)$ then $n \ll_s m$,
- if $\exists t \langle a, \mathcal{S} \rangle \sqsubseteq s$ such that $n \in \nu(a)$ and $m \notin \nu(t) \cup \nu(a)$ then $n \ll_s m$,
- if $\exists t \langle a, \mathcal{S} \rangle \sqsubseteq s$ such that $n, m \notin \nu(t) \cup \nu(a)$, $((n', n), (m', m)) \in \mathcal{S}$ and $n' \ll_s m'$ then $n \ll_s m$.

It is straightforward to show that \ll_s is a total (strict) ordering on $\nu(s)$. Note also that if $s \sqsubseteq t$ then \ll_t extends \ll_s . Recall that a strategy σ is *deterministic*

if $sa, sb \in \sigma$ implies $a = b$. We observe that a ν -strategy (or strategy-with-store) is deterministic if and only if it never introduces any new names — i.e. for all $s \in \sigma$, $P_\nu(s) = \emptyset$.

Lemma 5.5 *Given any compact strategy $\sigma : I \rightarrow \llbracket T \rrbracket$, there exists $k \in \mathbb{N}$, and a deterministic strategy $\underline{\sigma} : N^k \rightarrow \llbracket T \rrbracket$ such that $\mathbf{new}^k; \underline{\sigma} = \sigma$.*

PROOF: For each compact strategy σ , let $k = \max\{|P_\nu(s)| \mid s \in \sigma\}$ (which is well-defined, since σ contains only finitely many \sim -equivalence classes, and $s \sim t$ implies $|P_\nu(s)| = |P_\nu(t)|$). We derive $\underline{\sigma}$ from σ by replacing the i th name in each sequence $s \in \sigma$ (in the order \ll) with the i th name supplied by the first move.

For any sequence-with store s over $\llbracket T \rrbracket$, we may enumerate $P_\nu(s)$ as n_1, \dots, n_l (where $l \leq k$) using the ordering \ll_s . Given distinct names i_1, \dots, i_k , let s^{i_1, \dots, i_k} be the sequence-with store over $\llbracket T \rrbracket$ obtained by replacing n_j with i_j (i.e. we apply the automorphism f , where f is the finite partial injection which sends n_j to i_j for each $j \leq k$). So we may define $\underline{\sigma} = \{\langle i_1, \dots, i_k \rangle s^{i_1, \dots, i_k} \mid qs \in \sigma \wedge \forall j, j' \leq k. i_j = i_{j'} \implies j = j'\}$. \square

Hence if $\Lambda^k(\underline{\sigma})$ is definable as M then σ is definable as $M \mathbf{new} \dots \mathbf{new}$.

We now show that we may factorize out all *explicit* use of the store from strategies in our model. We first note that using the total ordering \ll_s on $\nu(s)$ we may uniquely represent any move occurring in a sequence-with-store s as a pair $\langle a, \mathcal{S} \rangle$, where \mathcal{S} is a *sequence* of pairs $(i_1, j_1) \dots (i_n, j_n)$, with $i_k \ll_s i_{k+1}$ for each $k < n$.

We say that a strategy $\sigma : A$ is *write-independent* if $\sigma \subseteq \text{WI}_A$ (so if σ is deterministic then $s \langle a, \mathcal{S} \rangle \langle b, \mathcal{S}' \rangle \in \sigma$ implies $\mathcal{S} = \mathcal{S}'$).

Lemma 5.6 *For any deterministic compact strategy $\sigma : I \rightarrow \llbracket T \rrbracket$ there exists a deterministic, compact write-independent strategy $\tilde{\sigma} : (N \rightarrow N \rightarrow I + I) \rightarrow \llbracket T \rrbracket$ such that $\Lambda^2(\mathbf{ass}); \tilde{\sigma} = \sigma$.*

PROOF: We represent the moves of the ν -arena $(N \rightarrow N \rightarrow I + I)$, in enabling order, as $\{\mathbf{write}(i) \mid i \in \mathbb{N}\}$, $\mathbf{ok1}$, $\{\mathbf{val}(j) \mid j \in \mathbb{N}\}$ and $\mathbf{ok2}_{\text{tt}}, \mathbf{ok2}_{\text{ff}}$. We derive $\tilde{\sigma}$ from σ by converting implicit assignments (i, j) in the store into explicit ones (sequences of moves $\mathbf{write}(i)\mathbf{ok1}\mathbf{val}(j)\mathbf{ok2}_{\text{tt}}$ in the argument $N \rightarrow N \rightarrow I + I$). Given a store \mathcal{S} , let $[i \mapsto j]_{\mathcal{S}}$ be the sequence: $\langle \mathbf{write}(i_1), \mathcal{S} \rangle \langle \mathbf{ok1}, \mathcal{S} \rangle \langle \mathbf{write}(j_1), \mathcal{S} \rangle \langle \mathbf{ok2}_{\text{tt}}, \mathcal{S}[i \mapsto j] \rangle$.

For each even-length $s \in L_A^S$, we define $\tilde{s} \in L_{(N \rightarrow N \rightarrow I + I) \rightarrow A}^S$ as follows: $\tilde{\varepsilon} = \varepsilon$, and if $\mathcal{S}' = ((i_1, j_1), \dots, (i_n, j_n))$ then $s \langle a, \mathcal{S} \rangle \langle b, \mathcal{S}' \rangle = \tilde{s} \langle a, \mathcal{S} \rangle [i_1 \mapsto j_1]_{\mathcal{S}} \dots [i_n \mapsto j_n]_{(i_1, j_1), \dots, (i_{n-1}, j_{n-1})} \langle b, \mathcal{S}' \rangle$.

$\tilde{\sigma} = \{\tilde{s} \mid s \in \sigma\}$ is deterministic, compact and write-independent, and $\Lambda(\text{ass}); \tilde{\sigma} = \sigma$ as required. \square

So if $\Lambda(\tilde{\sigma})$ is definable as $M : (\nu \Rightarrow \nu \Rightarrow o) \Rightarrow T$ then σ is definable as $M \lambda xy.x := y$.

Lemma 5.7 *Let $\sigma : I \rightarrow \llbracket T \rrbracket$ be a compact, write-independent strategy. Then there exists a compact $\nu\mathcal{G}$ strategy $\hat{\sigma} : (N \rightarrow N) \rightarrow \llbracket T \rrbracket$ such that $\sigma = \Lambda(\text{deref}); J(\hat{\sigma})$.*

PROOF: We derive $\hat{\sigma}$ from σ by explicitly dereferencing all of the locations in the store (by adding pairs of moves $\text{read}(i), \text{retn}(j)$ in $N \rightarrow N$) before each move by σ . For each sequence $s \in \text{WI}_{I \rightarrow \llbracket T \rrbracket}$ we define a sequence $\hat{s} \in L_{(N \rightarrow N) \rightarrow \llbracket T \rrbracket}$ as follows: $\hat{\varepsilon} = \varepsilon$, and
 $s \langle a, \mathcal{S} \rangle \langle b, \mathcal{S} \rangle = \hat{t} \text{aread}(i_1) \text{retn}(j_1) \dots \text{read}(i_n) \text{retn}(j_n) b$,
if $\mathcal{S} = (i_1, j_1) \dots (i_n, j_n)$. \square

So if $\Lambda(\hat{\sigma})$ is definable as $M : (\nu \Rightarrow \nu) \Rightarrow T$ then σ is definable as $M \lambda x.!x$.

We now use a further factorization to eliminate *implicit* use of the store: we reduce the compact, deterministic and store-independent strategies to finitary *innocent* strategies, by keeping an encoding of the history of play in the store. A similar factorization of a “knowing” strategy into the composition of an innocent strategy and a reference cell was originally described by Abramsky and McCusker [3] — the main difference in this case is that it is not possible to encode legal sequences as natural numbers, since names must be stored explicitly.

A deterministic strategy σ is *innocent* if its behaviour is always determined by the Player view (Definition 3.1) — i.e. if $sb, t \in \sigma$, ta is a legal sequence and $\ulcorner s \urcorner = \ulcorner ta \urcorner$ then $tab \in \sigma$. An innocent strategy is finitary if the set of orbits $\{\ulcorner s \urcorner \mid s \in \sigma\}$ is finite.

Lemma 5.8 *Let $\sigma : I \rightarrow \llbracket T \rrbracket$ be a deterministic, finitary strategy in $\nu\mathcal{G}$. Then there exists $j \in \mathbb{N}$, and a finitary, innocent strategy-with-store $\bar{\sigma} : N^j \rightarrow B$ such that $J(\sigma) = \text{new}^j; \bar{\sigma}$.*

PROOF: We assume that the length of sequences in σ is bounded above by $n \in \mathbb{N}$. Since G -action partitions the moves of $\llbracket T \rrbracket$ into a finite set of orbits we may assume an encoding $\phi : M_{\llbracket T \rrbracket} \rightarrow \mathbb{N}$ of these orbits such that $\phi(m) = \phi(m')$ iff $m \sim m'$, and $\phi(m) \leq l$ for all m . We may thus represent each move a_i in a justified sequence $s = a_1 \dots a_k$ over $\llbracket T \rrbracket$ using three values: $\phi(a_i)$, $\nu(a_i)$ and the target of its justification pointer $j(a_i)$. Thus we encode s as a store \mathcal{S}_s by storing these values at distinct locations mv_i , sp_i and js_i for each $i \leq k$, and adding a marker at mv_{k+1} that the sequence is complete — i.e. $\mathcal{S}_s = \{(mv_i, m_\phi(a_i)) \mid i \leq k\} \cup \{(mv_{k+1}, mv_{k+1})\} \cup \{(sp_i, \nu(a_i)) \mid i \leq k\}$

$k \wedge \nu(m_i) \neq \emptyset\} \cup \{(js_i, (j(a_i)) \mid 1 < i \leq k\}$.

For each non-empty even-length sequence $s \in L_{I \rightarrow \llbracket T \rrbracket}$ we define a set of sequences-with-store \bar{s} over $A = N^l \odot N^{n+1} \odot N^n \odot N^n \rightarrow \llbracket T \rrbracket$ as follows:

$\overline{qa} = \{\langle (m_1, \dots, m_l, mv_1, \dots, mv_{n+1}, sp_1, \dots, sp_n, js_1, \dots, js_n), \mathcal{S} \rangle \langle a, \mathcal{S} \rangle\}$
 $\overline{qasbc} = \{t \langle b, \mathcal{S} \rangle \langle c, \mathcal{S}[\mathcal{S}_{qasb}] \rangle \in L_A^S \mid t \in \overline{qa\bar{s}}\}$, where $\mathcal{S}[\mathcal{S}'] = \mathcal{S}' \cup \{(i, j) \in \mathcal{S} \mid i \notin \text{dom}(\mathcal{S}')\}$.

If $s \neq t$ then for any $s' \in \bar{s}$ and $t' \in \bar{t}$, $\ulcorner s' \urcorner \neq \ulcorner t' \urcorner$. Hence $\bar{\sigma} = \{\varepsilon\} \cup \{\bar{s} \mid s \in \sigma\}$ is innocent, and $(\text{new}^l \odot \text{new}^{n+1} \odot \text{new}^n \odot \text{new}^n); \bar{\sigma} = \bar{s}$ as required. \square

So if $\Lambda^{l+3n+1}(\bar{\sigma})$ is definable as M then σ is definable as $M \text{ new} \dots \text{new}$.

Since the strategy $\bar{\sigma}$ is not store-independent, we need to re-apply the factorizations of write-independence and read-independence, for which we require the following facts.

Lemma 5.9 *If σ is finitary and innocent, then the strategies $\tilde{\sigma}$ and $\hat{\sigma}$ are finitary and innocent.*

PROOF: We observe that if a given move-with-store occurs in the view of s , then the explicit assignment and dereferencing moves associated with that store occur in $\ulcorner \tilde{s} \urcorner$ and $\ulcorner \hat{s} \urcorner$. \square

Now we prove definability of the finitary innocent and store-independent strategies.

Lemma 5.10 *Given a finitary innocent store-independent strategy $\sigma : \llbracket \Gamma \rrbracket \rightarrow \llbracket T \rrbracket$, there exists a term of $\lambda \nu!_{\perp} - \{\text{new}, \text{ass}, \text{deref}\}$, $\Gamma \vdash M_{\sigma} : T$ such that $\llbracket M_{\sigma} \rrbracket = \sigma$.*

PROOF: We prove this using a decomposition of innocent strategies which closely follows Honda and Yoshida's proof of finite definability for innocent strategies in their model of call-by-value PCF [11]. The only difference is the presence of names as atomic datatypes on which the only definable operations are copying and equality-testing. So we give a sketch of the decomposition, which is by induction on the number of view-orbits in σ . We assume $\Gamma = x_1 : \nu, \dots, x_m : \nu, y_1 : o, \dots, y_n : o, z_1 : S_1, \dots, z_k : S_n$, where S_1, \dots, S_k are function-types.

If $\sigma = \perp$, then $M_{\sigma} = \Omega$. If $\sigma \neq \perp$, then it contains a view of the form $qa = \langle i_1, \dots, i_m, b_1, \dots, b_n, q_1, \dots, q_l \rangle a$. Let $\text{test}(j, k)$ be the term $x_j = x_k$ if $i_j = i_k$ and $\neg(x_j = x_k)$ otherwise. Then $M_{\sigma} =$

$\text{If } (\bigwedge_{j, k \leq m} \text{test}(j, k) \wedge \bigwedge_{j \leq n} (y_j = b_j)) \text{ then } M_{\sigma_{qa}} \text{ else } M_{\sigma^{qa}},$

where $\sigma^{qa} = \{s \in \sigma \mid qa \not\preceq s\}$ is smaller than σ , and therefore definable, and $\sigma_{qa} = \{\varepsilon\} \cup \{s \in \sigma \mid qb \preceq s\}$

We define $M_{\sigma_{qb}}$ by considering the possible form of the move b . If this is an answer to q — i.e. an initial move in $\llbracket T \rrbracket$ then:

- if $T = \nu$, then $b = i_j$ for some $j \leq m$, and we have $M_{\sigma_{qb}} = x_j$.
- if $T = T_1 \Rightarrow T_2$ then by uncurrying we obtain $\Lambda^{-1}(\sigma_{qb}) : \llbracket \Gamma \rrbracket \odot \llbracket T_1 \rrbracket \rightarrow \llbracket T_2 \rrbracket$ which is smaller, and hence definable by hypothesis, and so we let $M_{\sigma_{qb}} = \lambda x. M_{\Lambda^{-1}(\sigma_{qb})}$

If b is a move in $S_j = U \Rightarrow V$, then we may derive smaller strategies $\sigma_1 : \llbracket \Gamma \rrbracket \rightarrow \llbracket U \rrbracket$ (by relabelling views from σ_{qb} in which the question b has not been answered) and $\sigma_2 : \llbracket \Gamma, x : V \rrbracket \rightarrow \llbracket T \rrbracket$ (by removing the question b and its answer from views in σ_{qb} in which both occur, and relabelling). We may then define $M_{\sigma_{qb}} = (\lambda x. M_{\sigma_2})(z_j M_{\sigma_1})$. \square

Putting together the factorizations (Lemmas 5.5, 5.6, 5.7, 5.8 and 5.9) with the definability result for innocent strategies (Lemma 5.10) we have:

Proposition 5.11 *Every compact strategy $\sigma : I \rightarrow \llbracket T \rrbracket$ is definable as a term of $\lambda\nu!_{\perp}$.*

We may thus give a full abstraction result for the interpretation of terms as sets of complete sequences: a legal sequence s is complete if every question in s has been answered. We write $\llbracket M \rrbracket_c$ for the restriction of $\llbracket M \rrbracket$ to complete sequences. Note that we may define strategies as sets of complete plays and so define our fully abstract model of $\lambda\nu!$ directly (see [3]).

Proposition 5.12 *For all $\lambda\nu!_{\perp}$ -terms M, N , $M \lesssim N$ if and only if $\llbracket M \rrbracket_c \subseteq \llbracket N \rrbracket_c$.*

PROOF: This follows [3] etc. We sketch the proof for closed terms which easily generalizes to arbitrary terms in context.

Suppose $M \not\lesssim N$ — i.e. there exists $L : T \Rightarrow o$ such that $L M \Downarrow$ and $L N \not\Downarrow$, and so $\llbracket L M \rrbracket \neq \perp$ and $\llbracket L N \rrbracket = \perp$ by soundness and adequacy. So there exists a complete play $sa \in \llbracket L \rrbracket$ such that $qs \in \llbracket M \rrbracket_c$ and $qs \notin \llbracket N \rrbracket_c$.

Suppose $\llbracket M \rrbracket \not\subseteq \llbracket N \rrbracket$. Then there exists a complete play $qs \in \llbracket M \rrbracket$ such that $qs \notin \llbracket N \rrbracket$. By Proposition 5.11 the compact strategy on $\llbracket T \rrbracket \rightarrow I + I$ consisting of prefixes of complete plays $\{t \in L_{\llbracket T \rrbracket \rightarrow I+I}^S \mid t \preceq \mathbf{stt}\}$ is definable as a term $P : T \Rightarrow o$ such that $\llbracket P N \rrbracket = \perp$ and $\llbracket P M \rrbracket = \llbracket \mathbf{tt} \rrbracket$ and hence by adequacy $P M \Downarrow$ and $P N \not\Downarrow$ — i.e. $M \not\lesssim N$ as required. \square

By conservativity of $\lambda\nu!_{\perp}$ -equivalence over $\lambda\nu!$ -equivalence (Proposition 5.3) we have now shown full abstraction for $\lambda\nu!$.

Theorem 5.13 (Full abstraction) *For all $\lambda\nu!$ -terms M, N , $M \approx N$ if and*

only if $\llbracket M \rrbracket_c = \llbracket N \rrbracket_c$.

Using the fact that (by definition) for any term M of the nu-calculus, $\llbracket M \rrbracket^{\nu\mathcal{G}^s} = J(\llbracket M \rrbracket^{\nu\mathcal{G}})$, we may also observe that denotational equivalence in our simple model of the nu-calculus in $\nu\mathcal{G}$ precisely reflects observational equivalence in $\lambda\nu!$.

Corollary 5.14 *For any terms M, N of the nu-calculus, $\llbracket M \rrbracket_c^{\nu\mathcal{G}} = \llbracket N \rrbracket_c^{\nu\mathcal{G}}$ if and only if $M \approx N$ in $\lambda\nu!$.*

6 Conclusions and Further Directions

The most obvious remaining open problems relating to our model of $\lambda\nu!$ itself concern the extent to which it can be used to reason about observational equivalence and associated properties. Recent research (e.g. [9,23]) has succeeded in associating games models of finitary fragments of Idealized Algol with various classes of formal languages, leading to decidability results for observational equivalence. A natural next step would be to conduct a similar analysis for our model of $\lambda\nu!$. In order to characterize a finite-state fragment (for example) it would be necessary to restrict the ability to generate unbounded sets of new names. In general, observational equivalence in $\lambda\nu!$ is not decidable, because for types generated from $\{o, \Rightarrow\}$ it is conservative over equivalence of finitary Reduced ML, for which undecidability of observational equivalence (at second order) has been shown by Murawski [22].

A natural application of any model-checking analysis of our semantics would be to the verification of safety and information-flow properties in a variety of pointer-based models. We have mentioned that secrecy and access control problems can be described within our language $\lambda\nu!$, but we may also develop more precisely tailored semantics of languages in which names are used as keys, passwords or other secrets.

The model described here will hopefully form the basis of a general approach to the games semantics of a full range of named features, including references, objects in a heap, exceptions and channels. For example, we may construct a model of Reduced ML simply by storing integers at locations rather than names. However, the problem of determining which elements are definable (let alone the intrinsic equivalence) in this model is surprisingly involved: for example, a definable strategy cannot directly compare a name which has just been played to one which is outside the view (because names cannot be stored in Reduced ML) but might be able to do so indirectly. In one step in this direction, ν -arenas have been used to construct a game semantics of higher-order-concurrency, in which both names and processes may be passed

on named channels: we allow interaction on a channel as moves tagged with the channel name.

References

- [1] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation*, 148(1):1–70, 1999.
- [2] S. Abramsky, D. R. Ghica, A. S. Murawski, C.-H. L. Ong, and I. Stark. Nominal games and full abstraction for the nu-calculus. In *Proceedings of LICS '04*. IEEE Press, 2004.
- [3] S. Abramsky and G. McCusker. Linearity, Sharing and State: a fully abstract game semantics for Idealized Algol with active expressions. In P.W. O’Hearn and R. Tennent, editors, *Algol-like languages*. Birkhauser, 1997.
- [4] S. Abramsky and G. McCusker. Call-by-value games. In M. Nielsen and W. Thomas, editors, *Proceedings of CSL '97*, pages 1–17. Springer-Verlag, 1998.
- [5] S. Abramsky, K. Honda, G. McCusker. A fully abstract games semantics for general references. In *Proceedings of the 13th Annual Symposium on Logic In Computer Science, LICS '98*, 1998.
- [6] S. Abramsky, R. Jagadeesan. Games and full completeness for multiplicative linear logic. *Journal of Symbolic Logic*, 59:543–574, 1994.
- [7] S. Abramsky, R. Jagadeesan and P. Malacaria. Full abstraction for PCF. *Information and Computation*, 163:409–470, 2000.
- [8] M. J. Gabbay and A. M. Pitts. A new approach to abstract syntax with variable binding. *Formal Aspect of Computing*, pages 341 – 363, 2002.
- [9] D. Ghica and G. McCusker. The regular language semantics of second-order Idealised Algol. *Theoretical Computer Science*, 309:469 – 502, 2003.
- [10] R. Harmer and G. McCusker. A fully abstract games semantics for finite non-determinism. In *Proceedings of the Fourteenth Annual Symposium on Logic in Computer Science, LICS '99*. IEEE Computer Society Press, 1998.
- [11] K. Honda and N. Yoshida. Game theoretic analysis of call-by-value computation. In *Proceedings of ICALP '97*, volume 1256 of *Lecture Notes in Computer Science*. Springer-Verlag, 1997.
- [12] 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.
- [13] Alan Jeffrey and Julian Rathke. Towards a theory of bisimulation for local names. In *Proceedings of LICS '99*. IEEE Press, 1999.
- [14] J. Laird. *A Semantic Analysis of Control*. PhD thesis, Department of Computer Science, University of Edinburgh, 1998.

- [15] J. Laird. A fully abstract game semantics of local exceptions. In *Proceedings of LICS '01*. IEEE Computer Society Press, 2001.
- [16] J. Laird. A game semantics of ICSP. In *Proceedings of MFPS XVII*, number 45 in Electronic notes in Theoretical Computer Science. Elsevier, 2001.
- [17] J. Laird. A game semantics of local names and good variables. In *Proceedings of FOSSACS '04*, number 2987 in LNCS, pages 289–303. Springer, 2004.
- [18] J. Laird. A game semantics of higher-order concurrency. In S. Arun-Kumar and N. Garg, editors, *Proc. FSTTCS '06*, number 4337 in LNCS, pages 417–428. Springer, 2006.
- [19] O. Laurent. Polarized games. In *Proceedings of the Seventeenth International Symposium on Logic In Computer Science, LICS '02*, 2002.
- [20] G. McCusker. *Games and full abstraction for a functional metalanguage with recursive types*. PhD thesis, Imperial College London, 1996. Published by Cambridge University Press.
- [21] G. McCusker. On the semantics of the bad-variable constructor in Algol-like languages. In *Proceedings of MFPS XIX*, number 83 in ENTCS, 2003.
- [22] A. Murawski. On program equivalence in languages with ground-type references. In *Proceedings of LICS '03*. IEEE Press, 2003.
- [23] A. Murawski and I. Walukiewicz. Third-order Idealized Algol with iteration is decidable. In *Proceedings of FoSSACS '05*, number 3411 in LNCS, pages 202–218. Springer, 2005.
- [24] H. Nickau. Hereditarily sequential functionals. In *Proceedings of the Symposium on Logical Foundations of Computer Science: Logic at St. Petersburg*, LNCS. Springer-Verlag, 1994.
- [25] A. Pitts and I. Stark. Observable features of higher-order functions that dynamically create local names. In *Proceedings of MFCS '93*, pages 122–141, 1993.
- [26] J. Power and H. Thielecke. Environments in Freyd categories and κ -categories. In *Proceedings of ICALP '99*, number 1644 in LNCS. Springer, 1999.
- [27] J. Power and E. Robinson. Premonoidal categories and notions of computation. *Mathematical Structures in Computer Science*, 1997.
- [28] I. Stark. *Names and Higher-Order Functions*. PhD thesis, University of Cambridge Computer Laboratory, 1995.
- [29] I. Stark. Categorical models for local names. *Lisp and Symbolic Computation*, 1(9):122–141, 1996.
- [30] E. Sumii and B.C. Pierce. Logical relations for encryption. *Journal of Computer Security*, 11:521–554, 2002.